Software Concept and Documentation

PCS 8000

walter+bai ag

Testing Machines
Switzerland

Index

1. Introduction

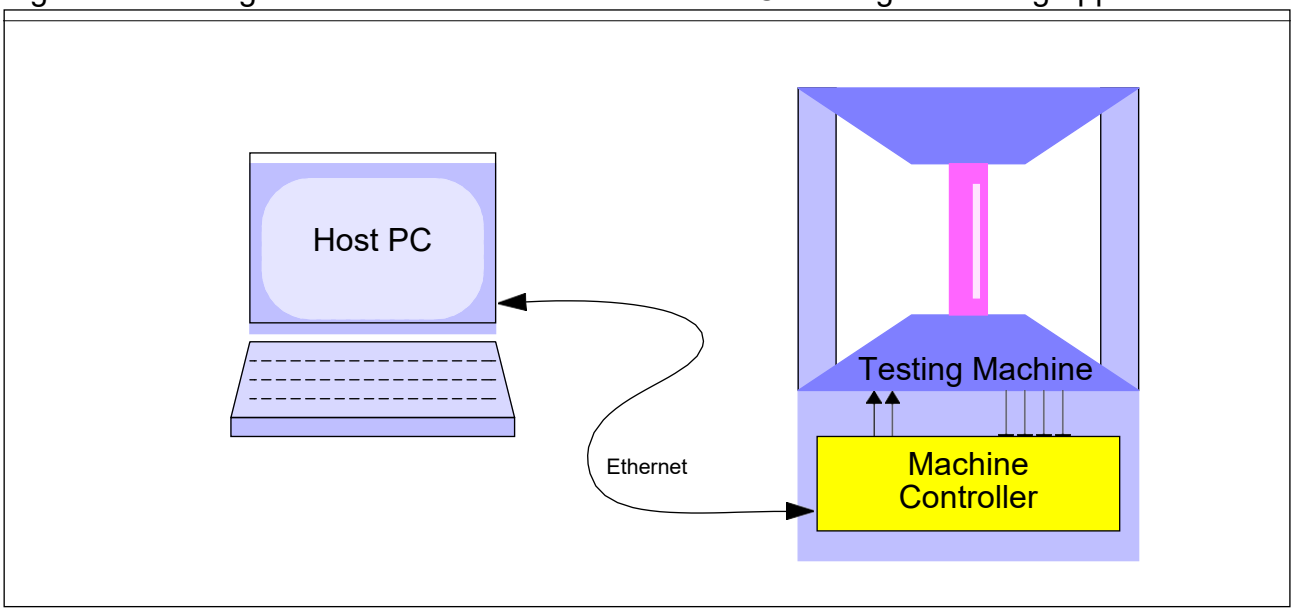
A testing environment consists of two major components (see figure 1):

1 The host PC.

2 The machine controller.

The host PC runs the user testing application. This is mainly the user interface. It provides input fields where the user can configure the testing machine and testing procedures. And it provides numerical and graphical displays showing measured or calculated data and testing results.
The machine controller manages all hardware, executes user commands on the machine and runs entire testing procedures. Its electronic hardware is equipped with a multitude of external interfaces for sensors, actors, the controlling host PC and others. Hence the controller is responsible for a lot of parallel tasks. Examples are the communication with the host PC, the acquisition of sensor data, the drive control loop, system supervision, and more. Some of these tasks, like the control loop, are time-critical and have to be handled in real-time with response times in the range of 20 µs. Therefor the controller's software is based on a hard real-time operating system.
Host PC and controller communicate over a standard ethernet connection using a unified XML based communication protocol. Among others, this protocol is described within this document (see chapter 10).

Figure 1: Testing machine with controller and host PC running the testing application.



2. Overview

The most important controller software modules are shown in figure 2. Some major groups may be identified:

- Central management and system database.
- External communication.
- System supervision.
- Sequencer.
- Data acquisition and control loop.

The central management with its system database is the controller's heart. It defines the controllers actual state and what commands and actions are allowed in this state. All external and internal com- mands are handled and dispatched by the central management. The system supervisor constantly checks whether the testing machine remains in a secure operating condition. If it detects a fault, an excessive stress or any operation beyond the limits it informs the central management which takes corresponding actions. The sequencer separates the control loop from the rest of the machine. It entirely masters the modules belonging to the control loop. It is able to execute whole testing sequences. The data acquisition subsystem is a highly sophisticated measurement and data process- ing unit. Among others it is able to exchange data streams with other controllers, to build virtual data streams, to generate events used in the sequencer, etc.
The major groups and modules are discussed in the following chapters.

3. Central Management

3.1. System states

At any time the testing machine is in a well defined system state. Every state may enable a set of pos- sible functions or tasks and may disable others. A state change may also trigger some internal tasks.

Startup

- This state is entered after system boot-up.
- Drive remains disabled.
- The start script is executed (see chapter 5.10).
- Initialization of all hardware and software modules.
- Self-diagnosis.
- Fill in system (sensor) database with initialization values and sensor data.
- In a multi-controller setup: establish connection to other controllers (see chapter 5.3).
- In single-controller setup or as master in a multi-controller setup: start listening for connec- tions from controlling host PC (see chapter 5.1).
- On success, the system state changes to standby. Otherwise it changes to error state.

Standby

- Wait for connection of controlling host PC.
- Establish group 4 communication links.
- The drive is disabled. The control loop is in off mode.
- Entering standby state disables the drive. It remains disabled while in standby state.

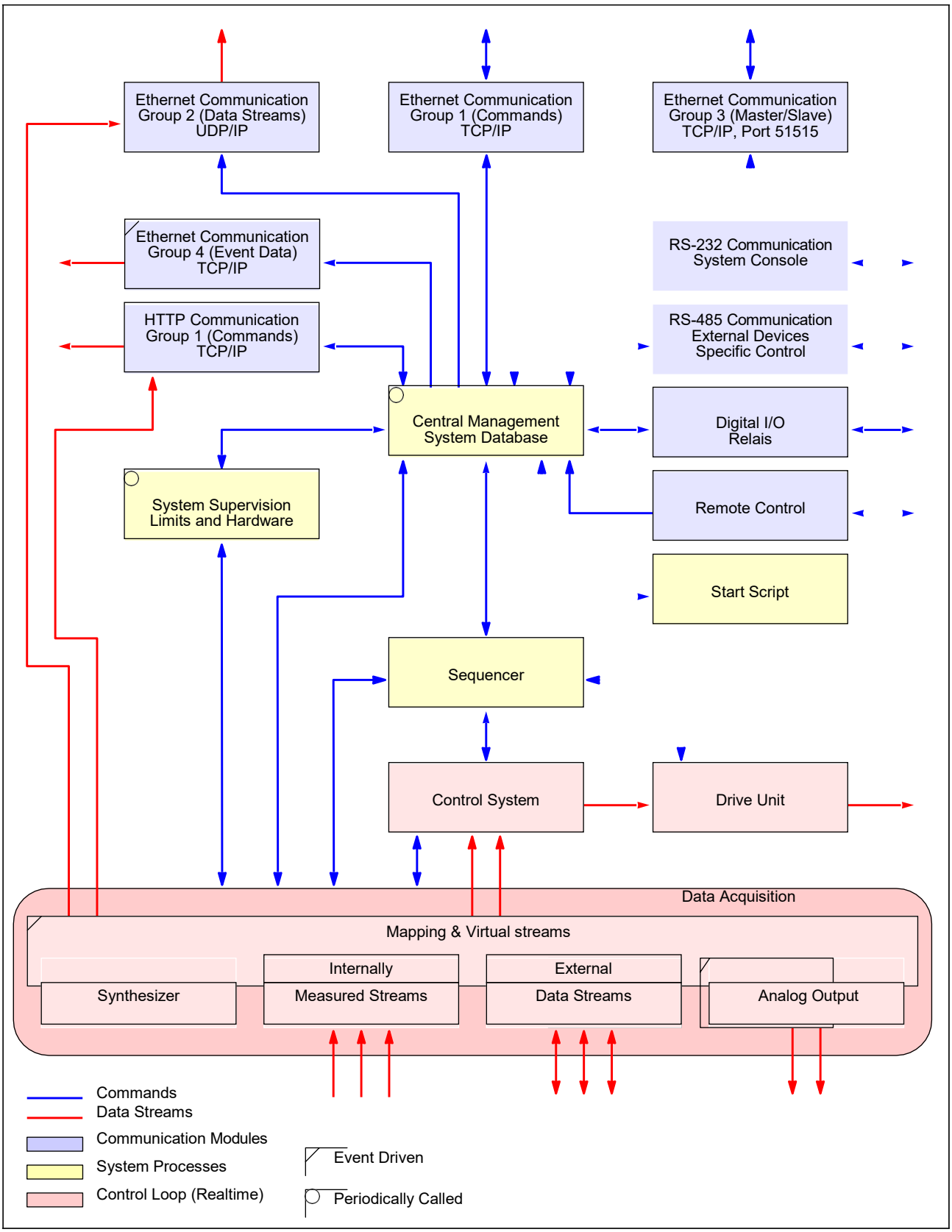


Figure 2:Software architecture, module overview.

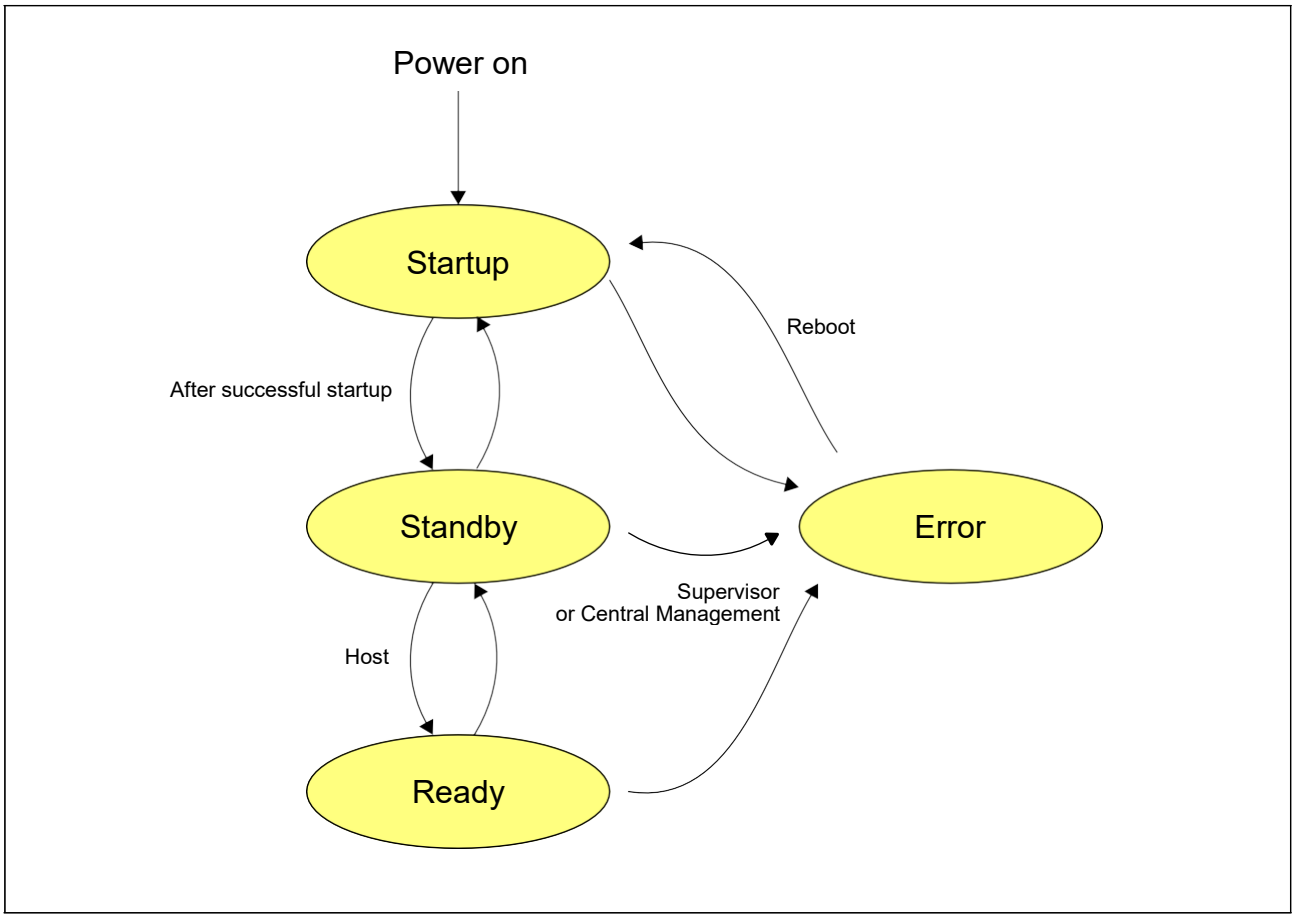


Figure 3: System states.

- Entering Standby state from Ready activates or reactivates the control loop in Setup mode for a certain period of time while the drive's power fades. After this period the control loop changes into Off mode.
- In a multi-controller setup the controller returns to startup when ever a master-slave connection gets lost.
- The controlling host PC can initiate a change into ready state.

Ready

- The testing machine is ready. The drive is enabled.
- Entering ready state enables the drive. The drive remains enabled while in ready state.
- Entering ready state activates or reactivates the control loop in setup mode. While in ready state the controlling host PC can change the control loop's mode (see chapter 7.1).
- The controlling host PC can initiate a change into standby state.

Error

- The testing machine and drive have been disabled after a critical error occurred.
- Entering error state disables the drive. It remains disabled while in error state.

- The only way to exit error state is through a system reboot.

3.2. Sensor Database

The sensor database stores important information about the controller's configuration and state. Table 1 lists the entries of the sensor database. Sensor details can be found in chapter 9. Other parts of the system database are managed by the corresponding software modules. A so-called thread data- base stores all information about threads (implementation of modules) including a pointer to the local section of the system database. Other modules can read that data by using the pointer from the thread database.

Field	Group	Description
version	Sensor format	EEPROM content version (int)[1].
check	Sensor format	CRC32 check sum over sensor data.
phys_type	Sensor type	Physical sensor type (list, see chapter 9).
elec_type	Sensor type	Electrical type of sensor (list, see chapter 9).
name	Sensor type	Name of sensor (string).
id	Sensor type	Identification, manufacturer's type (string).
sernum	Sensor type	Serial number (string).
description	Sensor type	Arbitrary description (string).
course sensitivity	Sensor parameters	Sensitivity (list, see chapter 9).
ex_volt	Sensor parameters	Excitation voltage (list, see chapter 9).
unit	Sensor parameters	Physical unit (string).
unit_id	Sensor parameters	Additional unit info (number, see chapter 9)
limit_behav	Sensor parameters	Behaviour at overload situation (list, see ch. 9).
adc_offset	Sensor parameters	Raw offset (int, in ADC steps).
limit_pos	Sensor parameters	Positive load limit (float, in physical unit).
limit_neg	Sensor parameters	Negative load limit (float, in physical unit).
pos_ovl_tol	Sensor parameters	Positive overload tolerance (float, in % of limit).
neg_ovl_tol	Sensor parameters	Negative overload tolerance (float, in % of limit).
resolution	Sensor parameters	Resolution for position sensors (double, in m/inc).
ssi_rate	Sensor parameters	SSI transfer speed (int)[2].
ssi_encoding	Sensor parameters	SSI encoding of bitstream (int, binary only).

Table 1: Sensor database.

Field	Group	Description
ssi_nof_bits	Sensor parameters	SSI number of bits (int)[3].
ipl_type	Sensor parameters	Interpolation type (int).
ipl_factor	Sensor parameters	Interpolation factors (int).
lin_type	Sensor linearization	Type of linearization (list, see chapter 9).
poly_order	Sensor linearization	Order of linearization polynomial (int).
n_ppos	Sensor linearization	Number of positive calibration points (int).
n_pneg	Sensor linearization	Number of negative calibration points (int).
ppos	Sensor linearization	Positive calibration points (array of float[4]).
pneg	Sensor linearization	Negative calibration points (array of float[5]).
polarity	Sensor parameters	Set input polarity (int, 1 or -1)
fine sensitivity	Sensor linearization	Sensitivity for linearization(float)
Nominal load	Sensor linearization	Nominal load in physical units (float)
filt_set	Sensor parameters	Low pass filter setting (3rd order IIR)[6]

Table 1: Sensor database.

1. The current version is 2 (since SW-version 0.9.2). 2.
250, 500, 750 or 1000kHz.
3. 8 to 32 Bits.
4. The array is an array of pairs of ADC values and corresponding physical values.
5. The array is an array of pairs of ADC values and corresponding physical values.
6. Cutoff frequencies available: 1500, 1000, 500, 250, 125, 60, 20Hz

3.3. Responsibilities

The central management is responsible for the following tasks:

- Change system states (Startup, Standby, Ready, Error).
- Enable or disable drive.
- Enable or disable clamp lock.
- System configuration handling (sensors, start script, see chapter 5.10)
- Supervisor exception handling (see chapter 6).
- Handle and/or dispatch all internal and external commands.

4. External Communication

Commands used over external communication interfaces are presented in chapter 10. This chapter outlines the general features of external communication links.

4.1. Ethernet Communication (Group 1)

The ethernet communication basics are outlined in [1]. In a multi-controller setup only the master controller exchanges group 1 statements with the controlling host PC. Slaves receive their commands over the group 3 interface. A group 1 communication always starts with request from the controlling host PC.
The group 1 software module is responsible for the following tasks:

- Open the TCP port, start server application, handle connections.
- Register clients in the system database (IP, rights)[1].
- Send and receive XML telegrams.
- Forward all XML telegrams addressed to a slave controller.
- Converts XML telegrams into internal representation.
- Re-synchronization after a transmission error or an XML error.
- Directly respond to connection check telegrams.
- Logging of connections and traffic..

Connection Parameters
According to [1] the following defaults are used:

- Host IP: 192.168.113.1.
- Master controller IP: 192.168.113.10.
- Protocol: TCP.
- Port number: 51512.
- Communication style: XML.

Client Registration
The process of setting up a TCP connection between the host PC (as client) and the master controller (as server) is as follows.

1 After booting the master controller starts listening on the mentioned TCP port.
The host PC sends a TCP connect request.

2 The controller affirms attendance by responding with a simple text string:
HELLO

1. So far only one client will be accepted. The acceptance of multiple clients might be realized in a later stage. As an alternative the web server interface can be used by multiple clients at the same time.

3 The host PC sends the following message string with coma but without spaces between tokens.
name,version,code

nameis a unique host PC name, for example “abacus61”.
versionis the supported protocol version. The actual version is “1.00”.
codeis a magic code, a kind of secret password.
The magic code can be set through the web interface or directly through the config file. For all parameters only alpha-numeric ASCII characters are allowed, no spaces, no special char- acters, no umlauts, no accents.

4 If everything is okay the controllers responds with the following.
OK

This opens a permanent TCP connection between host and master controller. Theconnection can be used to exchange XML statements as described in chapter 10.3. Note that even if aconnection has been established the controller keeps listening. If the controller does not accept the connection it responds with the following. Any further communication will then be rejected.
DENIED

Close a Connection
Under normal conditions the connection to the controlling host PC should never close.
When the controlling host PC (client) wishes to give up an existing connection, it simply closes the TCP socket. The controller detects the connection closing. It immediately stops all control loop activities if the lost connection belonged to the controlling host. This includes:

1 Quit any sequencer program and return to setup mode.
2 Turn off drive enable signal.
3 Wait for drive to fade and settle while still in position control.
4 Switch to system state ‘standby’.

During the entire process the controller keeps listening for new TCP connection requests.

Lost Connection
When the controller detects a lost connection to the controlling host PC, for example due to a broken cable, it behaves exactly the same as when the controlling host PC closes the connection (see above).
To detect a lost connection a polling mechanism is used. To enable this mechanism the <ecom_to> parameter in the config file must be set to a value greater than zero. The value reflects the timeout duration in seconds.

4.2. Ethernet Communication (Group 2)

See [1] for ethernet basics. The group 2 communication module does the following:

- Opens the UDP port.
- Receives commands to transmit data streams.
- Reads client data from the system database.
- Transmits data streams.
- Logging of commands and transmissions.
- Support for different logging levels: 'normal', 'debug', etc.

Communication Parameters

According to [1] the following defaults are used:

- Host IP: According to the requesting client.
- Controller IP: 192.168.113.10, 192.168.113.11, etc. according to the requested data stream.
- Protocol: UDP.
- Port number: 51513.
- Communication style: Binary packages, little endian.

Package Format

When multiple data streams are requested, the following principles apply:

- Each data stream uses its own packages.
- Approx. 10 packages per second and per stream are transmitted.
- In a multi-controller setup each controller sends its packages directly to the corresponding client.

The package format is as follows:

1. Data stream code (32 bit, unsigned int).
2. Master or slave controller number (32 bit, unsigned int).
3. Number of data objects in the package (32 bit, unsigned int).
4. Number of the data package (64 bit, unsigned int).
5. Time stamp of the first sample in the package (64 bit, unsigned int).
6. Minimum drag indicator at end of package (normally 32 bit, single precision float).
7. Maximum drag indicator at end of package (normally 32 bit, single precision float).
8. Data objects (normally 32 bit, single precision float each).

All package data are little endian[2]. Data stream codes are shown in table 2. Typically the number of data values per package is 800 at a sampling rate of 8 kHz and at 10 packages per second. Data pack- age numbers are consecutive numbers that help to reconstruct the data stream on the client side. Note that these numbers normally do not roll over as the counter period is thousands of years. The time stamp counts nanoseconds from the start of the controller. Again the time counter will hardly wrap around as the counter period is some 500 years. Data objects are packed one after the other, early samples first. The time stamp corresponds to the first value. The time interval between values directly relates to the system's sampling rate. Be aware that the time stamp counter and the package numbers might be reset concurrently under certain cirumstances, for example when sensor settings or configurations are changed. However, this will never happen while the control loop is running.

Stream	Data stream names	Data stream codes
Locally measured data streams	phys1 ... phys13	1 ... 13
Servo amplifier position stream[1]	phys14	81
Servo amplifier velocity stream[2]	phys15	82
Synthesizer output stream	synth	14
Dither output stream	dither	15
Virtual data streams	virt1 ... virt32	16 ... 47
Controller output stream	control	48
External data streams	ext1 ... ext16	49 ... 64
Calculated values	calc1 ... calc16	65 ... 80

Table 2: Data stream names and codes

1. Optional. Nur verfügbar wenn der Servoverstärker eingebaut ist.
2. Dito.

4.3. Ethernet Communication (Group 3)

In a multi-controller setup each controller knows from it's configuration if it is a master or a slave. If it is a slave it also knows its slave number. Slave numbers are shown in table 3.

Controller	Slave number
Master	0

Table 3: Slave numbers.

2. Conforming to all x86 architectures but opposing to IP standards.

Controller	Slave number
Slave number 1	1
Slave number 2	2
...	...

Table 3: Slave numbers.

The master controller knows how many slaves belong to the entire system. After startup the master waits for all slaves to connect. The system will only get ready when all slaves are connected.
The master controller's group 3 service implements a server which is responsible for the following:

- Open TCP port, wait for client connections.
- Register Clients and fill in system database.
- Send and receive XML telegrams.
- Forward XML telegrams to host PC if the telegram contains a slave number.
- Translate XML to internal command structure if the telegram lacks a slave number.
- Re-synchronization after an XML error.
- Directly respond to connection test telegrams.

A slave is responsible for:

- Connect at master's server application.
- Send and receive XML telegrams.
- Translate XML to internal command structure if the telegram lacks a slave number.
- Re-synchronization after an XML error.
- Periodically send connection test telegrams (link checking).

Connection Parameters
According to [1] the following defaults are used:

- Master controller IP: 192.168.113.10.
- Slave controller IP: 192.168.113.11,192.168.113.12, etc.
- Protocol: TCP.
- Port number: 51514.
- Communication style: XML.

Client Registration
The process of setting up a TCP connection between a slave controller (as client) and the master con- troller (as server) is as follows.

1 After booting the master controller starts listening on the mentioned TCP port.

2 The slave sends a TCP connect request.

3 The controller affirms attendance by responding with a simple text string:
HELLO

4 The slave sends the following message string with coma but without spaces between tokens.
number,version,code

numberis a unique slave number (see table 3).
versionis the supported protocol version. The actual version is “1.00”.
codeis a magic code, a kind of secret password.
The magic code can be set through the web interface or directly through the config file. For all parameters only alpha-numeric ASCII characters are allowed, no spaces, no special char- acters, no umlauts, no accents.

5 If everything is okay the master controllers responds with the following.
OK

This opens a permanent TCP connection between master and slave controller. The connec- tion can be used to exchange group 3 XML statements as described in chapter 10.5. Note that even if a connection has been established the controller keeps listening. If the controller does not accept the connection it responds with the following. Any further communication will then be rejected.
DENIED

Close a Connection
Under normal conditions the master to slave connections should only close when turning off the entire system. If, however, a master to slave connection breaks down, the master and slave behave exactly the same as described in chapter 5.1.

4.4. Ethernet Communication (Group 4)

In group 4 communication the controlling host PC implements a server service and all controllers are slaves. After the group 1 communication has established, the master tells the Host-IP-address to all slaves. After that, each controller tries to establish a connection to the host.
The controller’s group 4 communication module does the following:

- Connect to the controlling host PC.
- Transmit event data to controlling host PC.
- Re-connect after a connection loss.

Communication Parameters
The following defaults are used:

- Host IP: Given by the group 1 connection
- Master controller IP: 192.168.113.10.
- Protocol: TCP.
- Port number: 51515.
- Communication style: Binary packages.

Package Format

1 Event code (32 bit, unsigned int).
2 Master or slave controller number (32 bit, unsigned int).
3 Time stamp of the event (64 bit, unsigned int).
4 Event value (normally 64 bit, double precision float).
5 Length of optional data field (32 bit, unsigned int).
6 Optional data (string, vector, etc.).

Event Codes
The event codes in table 4 are used in the event package to identify events.

Code	Event	Values
0	Connection test telegram	0
10	Change of system state	new state(1)
20	Change of sequencer state	sequencer state(2)
30	Change of sequencer variable	variable value(3)
40	Send user defined string to host	0
101	Digital output 1 has changed (drive enable)	0 (off), 1(on)
102	Digital output 2 has changed	0 (off), 1(on)
103	Digital output 3 has changed	0 (off), 1(on)
104	Digital output 4 has changed	0 (off), 1(on)
105	Digital output 5 has changed	0 (off), 1(on)
106	Digital output 6 has changed	0 (off), 1(on)
107	Digital output 7 has changed	0 (off), 1(on)

Table 4: Event codes.

Code	Event	Values
108	Digital output 8 has changed	0 (off), 1(on)
109	Digital output 9 has changed	0 (off), 1(on)
110	Digital output 10 has changed	0 (off), 1(on)
111	Digital output 11 has changed	0 (off), 1(on)
112	Digital output 12 has changed	0 (off), 1(on)
201	Digital input 1 has changed	0 (off), 1(on)
202	Digital input 2 has changed	0 (off), 1(on)
203	Digital input 3 has changed	0 (off), 1(on)
204	Digital input 4 has changed	0 (off), 1(on)
205	Digital input 5 has changed	0 (off), 1(on)
206	Digital input 6 has changed	0 (off), 1(on)
207	Digital input 7 has changed	0 (off), 1(on)
208	Digital input 8 has changed	0 (off), 1(on)
209	Digital input 9 has changed	0 (off), 1(on)
210	Digital input 10 has changed	0 (off), 1(on)
211	Digital input 11 has changed	0 (off), 1(on)
212	Digital input 12 has changed	0 (off), 1(on)
301	Synthesizer: Wave1 cycle counter has incremented	Cycle counter wave1
302	Synthesizer: Wave2 cycle counter has incremented	Cycle counter wave2
303	Synthesizer: Waveform complete	Completion flag.
304	Synthesizer: Curve cycle counter has incremented	Curve cycle counter
401	Synthesizer: Ramp parameter trimmed.	New parameter value[4]
402	Synthesizer: Wave1 parameter trimmed.	New parameter value[5]
403	Synthesizer: Wave2 parameter trimmed.	New parameter value[6]
404	Synthesizer: Wave3 parameter trimmed.	New parameter value[7]
501	PIDx Controller: Parameter has been trimmed.	New parameter value[8]
551	Trigger threshold value has been trimmed.	New threshold value[9]
600	Key switch changed state (Master only)	0 (run), 1(setup)[10]

Table 4: Event codes.

Code	Event	Values
100'000	Emergency stop	0 (off), 1(on)
100'001	Emergency loop	0 (off), 1(on)
100'002	Emergency stop supply (Master only)	0 (ok), 1 (error)
100'020	High speed data link not locked	0 (off), 1(on)
100'030	Overflow on remote control counter (hand wheel)	0 (off), 1(on)
100'031	Overflow on stroke counter (A/B decoder)	0 (off), 1(on)
100'040	Interpolator velocity error	0 (off), 1(on)
100'041	Interpolator amplifier error	0 (off), 1(on)
100'042	Interpolator offset error	0 (off), 1(on)
100'043	Interpolator gain error	0 (off), 1(on)
100'050	Watchdog triggered	0 (off), 1(on)
100'055	Key switch error	0 (ok), 1 (error)(11)
100'060	Overtemperature detected	0 (off), 1(on)
100'061	Valve current is not equal to requested deviation	0 (off), 1(on)
100'062	Positive control voltage fault	Error value
100'063	Negative control voltage fault	Error value
100'064	Asymmetric control voltage fault	Error value
100'070	3.3 V supply voltage out of range	0 (off), 1(on)
100'071	1.2 V supply voltage out of range	0 (off), 1(on)
100'072	5.0 V supply voltage out of range	0 (off), 1(on)
100'073	+13 V supply voltage out of range	0 (off), 1(on)
100'074	-13 V supply voltage out of range	0 (off), 1(on)
100'075	2.5 V reference voltage out of range	0 (off), 1(on)
100'100	Sensor value exceeds positive limit	Module number
100'101	Sensor value exceeds negative limit	Module number
100'102	Sensor ADC overflow	Module number
100'103	Module bridge voltage out of range (DMS, DC)	Module number
100'104	Module bridge current out or range (LVDT, DMS)	Module number

Table 4: Event codes.

Code	Event	Values
100’105	General module/sensor status error	Module number
100’150	Maximum physical probe limit exceeded	Module number
100’151	Minimum physical probe limit exceeded	Module number
100152	Maximum virtual probe limit exceeded	Channel number
100153	Minimum virtual probe limit exceeded	Channel number
100200	Valve drive output signal clipped	0
101000	Hydraulic module output failure(12)	Channel number
101001	Hydraulic module relais failure (broken relais)	0 (ok), 1 (error)
101002	Hydraulic module input power failure(13)	0 (ok), 1 (error)
101003	Hydraulic module emergency stop failure(14)	0 (ok), 1 (error)
102000	Tapping point system pressure failure	0 (ok), 1 (error)
102001	Tapping point output pressure failure(15)	0 (ok), 1 (error)
102002	Leak oil pump run failure(16)	1 (error)
103000	Traverse control, Clamp lock active (operation denied)	0
103001	Traverse control, pressure lost while moving	0
103010	Traverse unclamp procedure in progress	0: starting, no pressure 1: pressure reached
104000	Clamp control, Clamp lock active (operation denied)	0
104001	Clamp control, pressure state on upper clamp while closing	0: starting, no pressure 1: pressure reached, clamp closed
104002	Clamp control, pressure lost on upper clamp in closed state	0
104003	Clamp control, pressure state on lower clamp while closing	0: starting, no pressure 1: pressure reached, clamp closed

Table 4: Event codes.

Code	Event	Values
104004	Clamp control, pressure lost on lower clamp in closed state	0
104010	Clamp control, opening upper clamp in progress	0: starting, clamp closed 1: timer elapsed, clamp opened
104011	Clamp control, opening lower clamp in progress	0: starting, clamp closed 1: timer elapsed, clamp opened

Table 4: Event codes.

1. Program states: 0 = error state, 1 = startup state, 2 = standby state, 3 = ready state.
2. Sequencer states: 0 = off state, 1 = fade state, 2 = setup state, 3 = program state.
3. The variable name stands in the optional string field of the group 4 message.
4. The parameter name stands in the optional string field of the group 4 message.
5. The parameter name stands in the optional string field of the group 4 message.
6. The parameter name stands in the optional string field of the group 4 message.
7. The parameter name stands in the optional string field of the group 4 message.
8. The parameter name stands in the optional string field of the group 4 message.
9. The trigger name stands in the optional string field of the group 4 message.
10. The normal state is “run” when the switch is closed. On state “setup” no sequencer program can be started.
11. The key switch has two contacts, which both must be closed or opened simultaneously. If this is not the case, the controller will do an emergency stop and switch to startup mode.
12. ..Der Zustand eines Ausgangs stimmt nicht, verursacht evtl. durch Defekt, Kurzschluss, etc.
13. Die 24V Speisung aus dem separaten Netzteil fehlen.
14. Das zugeführte Notstopkreissignal stimmt nicht mit dem internen Zustand überein (evtl. Verdrahtungsfe- hler).
15. Das OK-Event wird erst gesendet, wenn der Regler im Ready-Zustand ist und das Drucksignal erscheint. Das Error-Event wird gesendet, wenn der Regler im Ready-Zustand ist und das Drucksignal über die ganze Zeitdauer des Timers T8 ausgefallen ist.
16. Das Error-Event wird gesendet, wenn der Regler im Ready-Zustand ist und die Rückmeldung der Pumpe während mehr als 5 Sekunden ausfällt.

Client Registration

The process of setting up a TCP connection between a controller (as client) and the host PC (as server) is as follows.

1 After booting the host PC starts listening on the mentioned TCP port.

2 The controller sends a TCP connect request.

3 The host PC affirms attendance by responding with a simple text string:

HELLO

4 The controller sends the following message string with coma but without spaces between tokens.

number,version,code

numberis a unique controller number (see table 3).
versionis the supported protocol version. The actual version is “1.00”.
codeis a magic code, a kind of secret password.
The magic code can be set through the web interface or directly through the config file. For all parameters only alpha-numeric ASCII characters are allowed, no spaces, no special char- acters, no umlauts, no accents.

5 If everything is okay the host PC responds with the following.

OK

This opens a permanent TCP connection between host PC and controller. The connection can be used to exchange group 4 event messages. Note that even if a connection has been estab- lished the host PC keeps listening for connections from other controllers in multi-controller setups. If the host PC does not accept the connection it responds with the following. Any fur- ther communication will then be rejected.
DENIED

Close a Connection
Under normal conditions the host to controller connections should only close when turning off the entire system. If, however, a connection breaks down, the host PC and controller behave exactly the same as described in chapter 5.1.

4.5. Ethernet Communication Imetrum Stream Protocol

Dies ist eine Implementation des TCP/IP Protokolls der Firma Imetrum und dient zum Anschluss von Videomesssystemen. Sie basiert auf dem Dokument “Video Gauge 5.2 - Socket Comms Proto- col” und unterstützt zur Zeit maximal 5 Messkanäle.
Die Messkanäle können einzeln zugeschaltet werden, wobei die Daten jeweils einem virtuellen Kanal zugewiesen werden. Jeder Kanal unterstützt einen Offset, sowie einen Skalierungsfaktor.Eine Interpolation zwischen zwei Messpunkten ist zuschaltbar. Die Datenverarbeitung erfolgt äquivalent zum DOLI-Protokoll, siehe Kapitel 5.6 RS-232 Communication.
Die Konfiguration dieses Dienstes erfolgt über ein XML-File, welches beim Starten gelesen wird. Das entsprechende Telegramm kann auch zur Laufzeit über die Gruppe1-Kommunikationsverbind- ung (siehe chapter 5.1) geschickt, resp. abgefragt werden.

4.6. RS-232 Communication

System console
The serial port is used as a system console. This access is reserved for use during development or debugging.

Login Parameter:

- User name: root
- Password: (none)

DOLI-Protokoll
Das offen gelegte serielle Protokoll von DOLI ist implementiert und zugänglich über die gleiche Schnittstelle. Sobald dieses Protokoll über das Webinterface aktiviert wird, wird die Systemkonsole abgeschaltet. Ein serieller Systemzugang ist dann nicht mehr möglich.
Das DOLI-Protokoll ermöglich die Abfrage von bis zu 4 Datenströmen von einem externen Gerät, z.B. Videoextensometer. Diese Daten können je einem virtuellen Kanal zugewiesen werden, wo sie dann für die Weiterverwendung zur Verfügung stehen. Diese Daten fallen asynchron und typischer weise mit einer wesentlich geringeren Abtastrate an, im Vergleich zum internen Regeltakt. Dies kann, insbesondere falls diese Daten zur Regelung verwendet werden sollen zu Problemen führen. Aus diesem Grund kann für jeden Kanal eine Interpolation zugeschaltet werden, welche zwischen den einzelnen Datenpunkten Werte mit Hilfe einer Cosinusfunktion interpoliert. Die zeitliche Länge der Interpolation beträgt 20% des zeitlichen Abstandes zwischen zwei Datenpunkten und passt sich adaptiv an.
Jeder der 4 Datenströme kann unabhängig ein- oder ausgeschaltet werden. Weiter kann jedem Daten- strom ein Skalierungs- und ein Offsetparameter zugewiesen werden.
Die Berechnung der Datenwerte erfolgt nach:
Datenwert = Rohwert * scale - offset;
Danach wird wie bei allen Messkanälen noch der Useroffset addiert und die Schleppzeiger nach- geführt.

4.7. HTTP Communication

The controller provides comfortable access to data and management features by web browser. The HTTP service can be enabled or disabled.

Access Parameters

- Port 80: Unrestricted access.

4.8. RS-485 Communication

The RS-485 port will be used for communication to external devices like conditioning cabinet, etc. The functions and protocol specifications depends on the particular device.

Eurotherm temperature controller
The system has support for Eurotherm temperature controllers series 2000. It supports up to ten con- currently connected devices. Below is a list of supported operations:

- Periodic readout of process value and provide it as a virtual stream.
- Synchronous read and write of operating parameters through direct XML commands.
- Asynchronous read and write operation from within sequencer programs.

- Assign an operating value to a sequencer variable (within programs only).
- Trigger generation after read or write operation for sequencer program synchronisation.
- Parameter setup and configuration through webinterface.

4.9. Digital I/O

Digital I/O lines are available to control or test external devices, switches or states. On the PCS8000- V1 there are 8 output and 12 input lines available. On the PCS8000-V2/V3 there are 12 output lines available. The optional hydraulic module (PCS8000.HSM) offers further 15 output lines (usable only with PCS8000-V3).

Assignment
Some I/O lines are assigned to dedicated functions, see tables 5 and 6.

Input Nr.	Function
1	Drive ready
2	Upper limit switch
3	Lower limit switch
4	Safety door ready
5 ... 12	General purpose

Table 5: Digital input assignment (valid only for V1 controller).

Output Nr.	Function
1	Drive enable
2	Clamp lock (V1 only)
3	Safety door lock (V1 only)
4 ... 8	General purpose (V1)
2 ... 12	General purpose (V2/V3)
HM1 ... HM15	General purpose on hydraulic module

Table 6: Digital output assignment.

Table 5 only applies to the PCS8000-V1 controller. On the PCS8000-V2/V3 only the “Drive enable”-output is permanently assigned. All other functions are freely configurable.

The digital input states are sampled in hardware (FPGA) periodically (100Hz). The operating sys- tem's driver recognizes input transitions in real-time and generates corresponding events. These events can be used to issue a message to the controlling host PC (see chapter 5.4) or to trigger actions in the sequencer.
The user only has access to the general purpose outputs. The other outputs are handled by the central management and the user can not overwrite them for security reasons.

Hydraulik-Modul und Zapfstellensteuerung (manifold control)

Das Hydraulikmodul bietet weitere 15 geschaltete digitale Ausgänge. Im Gegensatz zu obigen Aus- gängen handelt es sich aber nicht bloss um (Relais-) Kontakte, sondern um aktive 24V-Ausgänge. Diese werden intern von einem separaten Netzteil gespiesen, sind galvanisch getrennt und liefern bis zu insgesamt 180W Leistung. Ausserdem sind sie im Unterschied zu obigen Ausgängen in den Notauskreis integriert und schalten bei einem Notaus sicher ab. Weiter sind sie nicht mit herkömmli- chen Relais realisiert, sondern mit elektronischen Schaltern. Dies macht sie besonders geeignet für das Schalten von Ventilen (induktive Lasten).
Neben den geschalteten Ausgängen besitzt das Hydraulikmodul noch einen analogen Stromausgang, der bis zu 1A an 24V liefern kann und eine Ditherfunktion besitzt. Dieser Ausgang ist für die Kraft- steuerung der Spannköpfe vorgesehen.
Grundsätzlich lassen sich die geschalteten Ausgänge des Hydraulikmodules gleich verwenden wie die digitalen onboard-Ausgänge. Sie sind allerdings speziell konzipiert worden für die Ansteuerung der Ventile einer Zapfstelle. Die Steuerung der Ventile ist in die Firmware integriert und kann fol- gende Aktoren bedienen:

- "Füllen"-Ventil (Fill up)
- "Einrichten aus"-Ventil (Set up off)
- "Entlasten aus"-Ventil (Pressure relief off)
- "Leistung1..3"-Ventile (power1..3), simultan geschaltet
- Leckölpumpe (Leakoil_pump)

Aktoren, die geschaltet werden sollen, müssen in der IO-Konfiguration einem bestimmten Ausgang zugewiesen werden. Ein nicht zugewiesener Aktor wird nicht geschaltet. (siehe Kapitel 10.11).
Neben den Ausgängen sind auch drei digitale Eingänge bei der Zapfstellensteuerung involviert. Dort werden zwei Pressostaten angeschlossen. Einmal vor der Zapfstelle (system pressure) und einmal hinter der Zapfstelle (output pressure). Weiter kann ein Rückmeldesignal der Leckölpumpe ange- schlossen werden (Pumpe läuft). Auch die Eingänge müssen in der IO-Konfiguration zugewiesen werden und auch hier gilt: ein nicht zugewiesener Eingang gilt als nicht vorhanden und wird dementsprechend nicht beachtet.
Die Konfiguration der Zapfstellensteuerung (siehe Kapitel 10.22) bestimmt das zeitliche Verhalten der Steuerung. Abbildung 4 und Tabelle 7 zeigen die Funktion und die einstellbaren Zeiten:

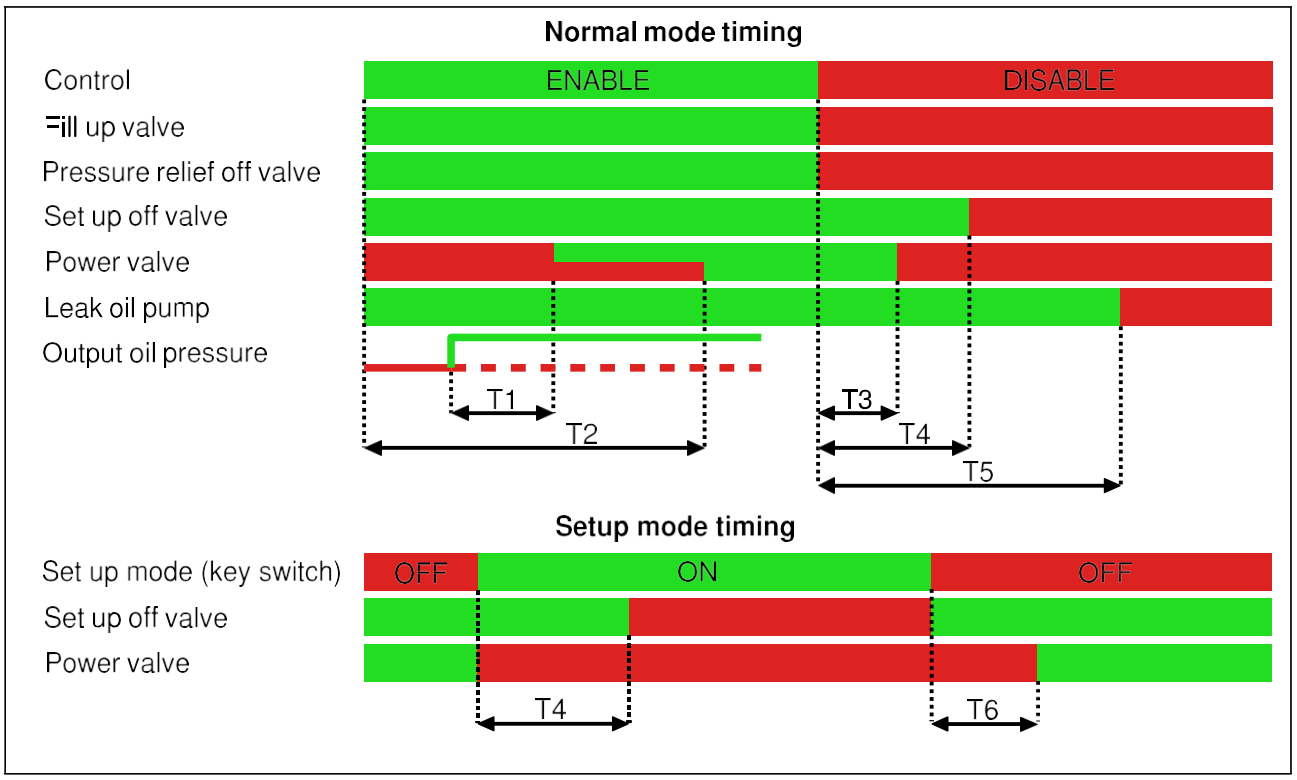


Figure 4: Tapping point timing

Zeiten	Bedeutung	Zulässiger Bereich (Sek)	Default Werte (Sek)
T1	Verzögerung Leistung Ein nachdem der Ausgangspres- sostat Druck sieht.	0 - 20	2
T2	Verzögerung Leistung Ein nachdem Regelung einge- schaltet (unabhängig von Pressostat)	0 - 20	5
T3	Verzögerung Leistung Aus nachdem Regelung ausge- schaltet.	0 - 5	0
T4	Verzögerung Einrichten nachdem Regelung ausge- schaltet oder nachdem Schlüsselschalter auf Normal- betrieb.	0 - 10	5
T5	Verzögerung Leckölpumpe Aus nachdem Regelung aus.	0 - 7200	600

Table 7: Einstellbare Zeitwerte der Zapfstellensteuerung

Zeiten	Bedeutung	Zulässiger Bereich (Sek)	Default Werte (Sek)
T6	Leistung Ein nachdem Schlüsselschalter auf Normal- betrieb.	0 - 5	2
T7	Ueberwachung Systemdruck, zulässige Fehlerzeit. Beim Auftreten eines Druckabfalls wird nach dieser Zeit ein Fehlerzustand registriert, ein entsprechendes Event verschickt und die Maschine abgeschaltet.	0 - 15	5
T8	Ueberwachung Ausgangsdruck, zulässige Fehlerzeit.	0 - 15	5

Table 7: Einstellbare Zeitwerte der Zapfstellensteuerung

Die zulässige Fehlerzeit für das Rückmeldesignal der Pumpe ist nicht einstellbar und beträgt 5 Sek.

Traversensteuerung

Im Zusammenhang mit dem Hydraulikmodul ist auch die Traversensteuerung in die Firmware inte- griert. Folgende Ventile (Ausgänge) kommen dabei zur Anwendung:

- Querhaupt klemmen / entklemmen 1 (Traverse_clamp_valve1)
- Querhaupt klemmen / entklemmen 2 (Traverse_clamp_valve2)
- Aufwärts fahren (Traverse_up)
- Abwärts fahren (Traverse_down)

Die ersten beiden Ausgänge werden simultan geschalten. Ausserdem ist folgender digitaler Eingang benutzt:

- Druckschalter “Querhaupt Entspanndruck” (Pressure_traverse_clamp)

Obige Funktionen müssen in der IO-Konfiguration den gewünschten Ein- und Ausgängen zugewie- sen werden.
Folgende Zeiten sind bei der Traversensteuerung einstellbar:

Zeiten	Bedeutung	Zulässiger Bereich (Sek)	Default Werte (Sek)
T1	Zeitdauer für Vorfüllen bevor die Traverse entspannt wird	0 - 5	2.5
T2	Verzögerung nach Entspannung, bevor Auf oder Ab aktiv wird.	0 - 5	2.5

Table 8: Einstellbare Zeitwerte der Traversensteuerung

Spannkopfsteuerung

Im Zusammenhang mit dem Hydraulikmodul ist auch die Spannkopfsteuerung in die Firmware inte- griert. Folgende Ventile (Ausgänge) kommen dabei zur Anwendung:

- Oberen Spannkopf öffnen (Open_upper_clamp)
- Oberen Spannkopf schliessen (Close_upper_clamp)
- Unteren Spannkopf öffnen (Open_lower_clamp)
- Unteren Spannkopf schliessen (Close_lower_clamp)
- Oder-Verknüpfung aus obigen Operationen (Any_clamp_operation)

Das letzte Ventil spricht an, sobald mindestens eines der ersten vier Ventile anspricht. Ausserdem sind folgende digitalen Eingänge benutzt:

- Druckschalter oben (Upper_clamp_closed)
- Druckschalter unten (Lower_clamp_closed)

Obige Funktionen müssen in der IO-Konfiguration den gewünschten Ein- und Ausgängen zugewie- sen werden. Folgende Zeiten sind bei der Spannkopfsteuerung einstellbar. Beim Oeffnen der Spannköpfe werden die entsprechenden Ventile nach Ablauf dieser Zeit automatisch deaktiviert ohne dass ein Stop- Befehl geschickt werden muss.

Zeiten	Bedeutung	Zulässiger Bereich (Sek)	Default Werte (Sek)
T1	Oeffnungszeit oberer Spannkopf	0 - 60	30
T2	Oeffnungszeit unterer Spannkopf	0 - 60	30

Table 9: Einstellbare Zeitwerte der Spannkopfsteuerung

Spannkraftsteuerung

Um die Kraft der Spannköpfe zu steuern, besitzt das Hydraulikmodul einen proportionalen Ausgang, der von 0 bis 1A/24V liefern kann. Ausserdem kann ein Dithersignal[3] zugeschaltet werden.
Die Spannkraftsteuerung besitzt drei Modi: Aus, Manuell, Automatisch.
Im manuellen Modus wird ein fixes Ausgangssignal eingestellt. Dieses kann via Konfiguration oder via Fernsteuerung verstellt werden. Im automatischen Modus wird das Ausgangssignal linear mit der gemessenen Kraft[4] der Probe erhöht gemäss folgender Grafik:

3. Frequenz: 40...200 Hz; Amplitude: 0...50% der Gesamtaussteuerung.
4. Als Messkanal wird dafür der Clamplock-Kanal verwendet (siehe Systemkonfiguration in Kapitel 10.2).

Anmerkung: Um den Strom am proportionalen Ausgang regeln zu können, besitzt das Hydraulik- modul eine integrierte Strommessfunktion. Der physikalische Messkanal 3 (Stream phys3) liefert dabei die Stromwerte skaliert in Ampere.

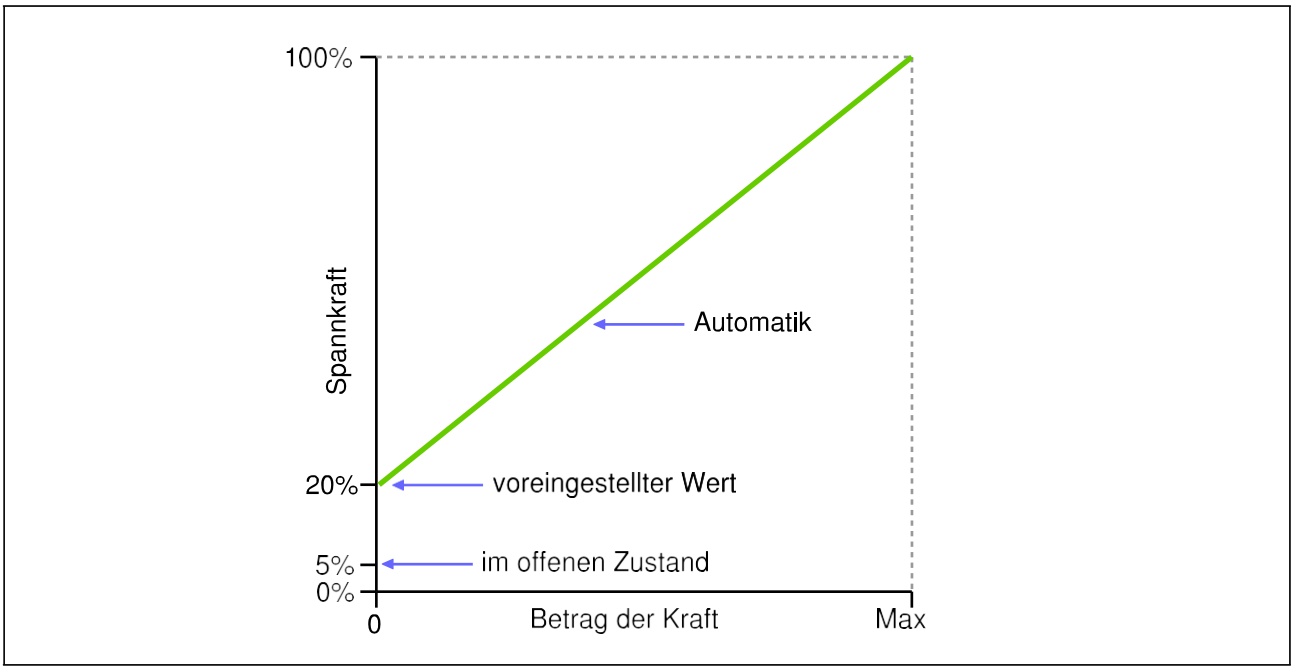


Figure 5: Automatische Spannkraftsteuerung

4.10. Start Script

At system startup the start script module reads XML commands from files (start.xml, config.xml) and feeds them into the central management. These commands are used to configure the system.

4.11. Remote Control

The controller is equipped with a remote control that offers some buttons, displays and a hand wheel. A limited set of operations can be called from the remote control. One example is the trimming of synthesizer or control loop parameters.

The controller hardware polls the remote control buttons and sends real-time events to the controller software whenever a button is pressed or released.

In a multi-controller setup only the master can have a remote control. Every controller configured as a slave disables its support for remote control hardware. However the master's remote control can be used for example to trim slave parameters.

The new remote controller is a graphical touch screen based solution. It communicates with the con- troller by a connectionless XML-based protocol on RS232 lines. The communication speed is 230400 Bit/s.

5. System Supervisor

The system supervisor ensures the reliable and safe operation of the controller.

Checkings

- Emergency stop switch and emergency stop loop in multi-controller setups.
- Internal supply voltages.
- Internal reference voltages.
- Valve current and drive output level.
- DMS bridge supply voltages and current.
- LVDT current.
- Input ADC range.
- Sensor limits.
- High speed data link connection (PLL locked).
- Controller temperature.
- Probe limits (physical and virtual data channels).
- Clamp lock.

Wherever a problem occurs the system supervisor sends a message to the central management. The central management is responsible for taking corresponding actions.

6. Sequencer

The synthesizer, control and drive module operate under control of the sequencer. There are basically two sequencer modes, 'setup' and 'program' (see chapter 7.1). In 'setup' mode, synthesizer,control- ler and drive module can be configured and started by hand. For security reasons default settings are used whenever the 'setup' mode is entered.
In 'program' mode entire experimental programs are conducted automatically.

6.1. Sequencer States

At any time the sequencer is in one of the states shown in figure 6. The sequencer can change from one state to another by itself or by an internal or external command.

Startup/Off

- Start state during and after power-on.
- The drive is disabled.
- The control loop is off.
- The drive unit outputs zero.
- Possible state change: setup state.

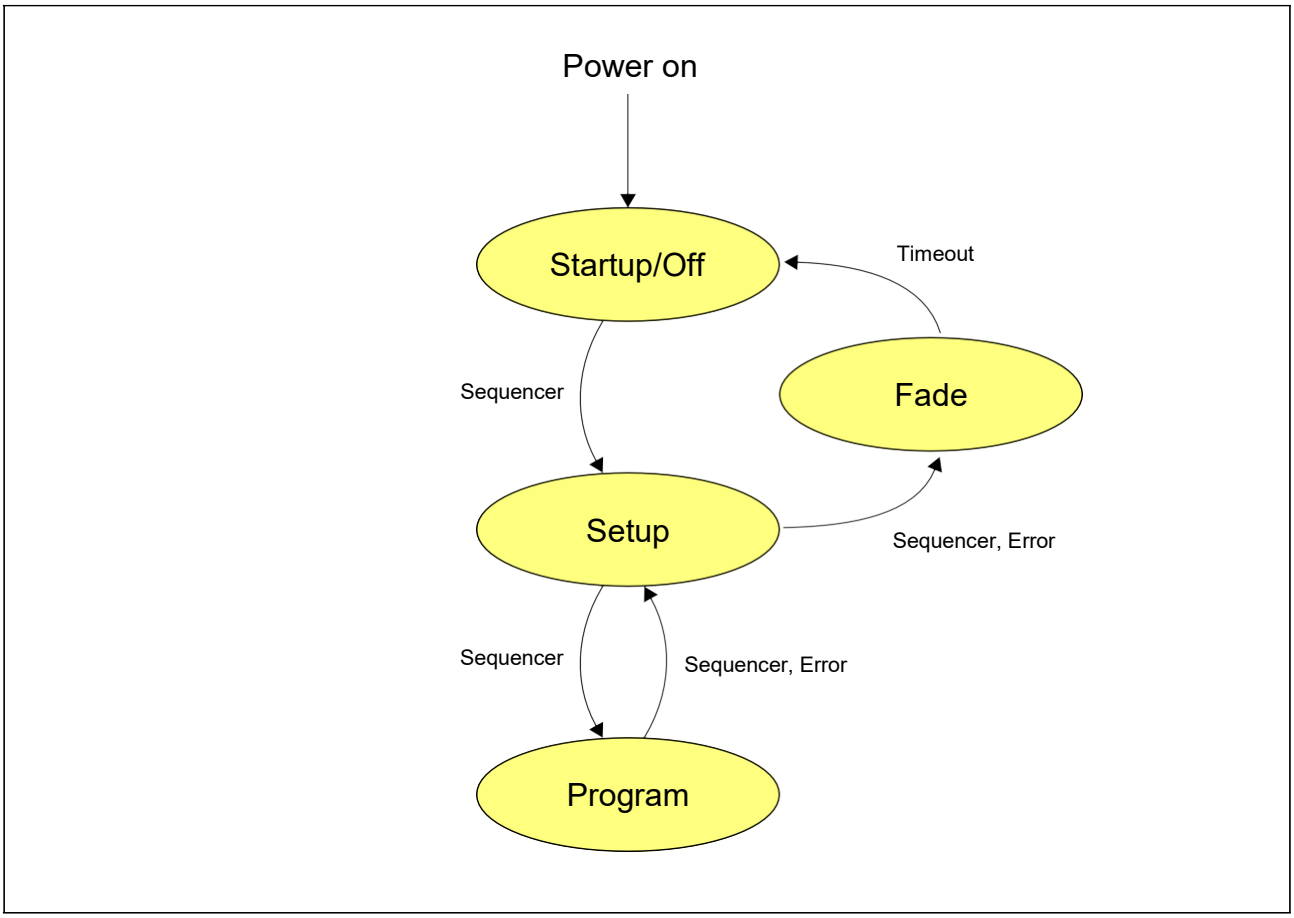


Figure 6: Sequencer states.

Setup

- This state is used to setup the machine.
- Whenever the 'setup' state is entered default parameters for the control loop are loaded. Normally a combined stroke-force controller will then be used.
- The user can not directly modify these default parameters because they are relevant for security.
- When in the 'setup' mode the user can overwrite the control loop parameters. For example he can choose another control strategy or he can use a different data stream to control or another synthesizer function. This may be useful for example to bring the machine into an certain position. However, the new settings will be lost when reentering 'setup' mode.
- The output values of the drive unit are limited in 'setup' mode. The limits are configured in the start script.
- The user can not directly modify these default limits because they are relevant for security.
- When in 'setup' mode the user can overwrite the drive limits to obtain more flexibility. However, the new settings will be lost when reentering 'setup' mode.
- Some synthesizer and control system parameters can be trimmed, for example by hand wheel on the remote control (see chapter 5.11) or by buttons on the host PC.
- In this state the user can manually release the machine out of an overload situation or if an

end position switch has been reached.

- Possible state changes: fade state, program state.

Fade

- Drive is disabled.
- Control loop remains active while drive fades.
- State change after timeout: off mode.

Program

- The control loop is under program control.
- Some synthesizer and control system parameters can be trimmed, for example by hand wheel on the remote control (see chapter 4.10) or by buttons on the host PC.
- Possible state change: setup state.

6.2. Sequencer Program

A program is a series of commands like

- configure the control loop (data acquisition, synthesizer, control system, drive unit),
- start the control loop,
- wait for an event (completion of synthesizer, limit reached, timeout, host, etc.),
- take some actions (switch a digital output or relays, send an RS-485 command, etc.),
- exchange data with host PC,
- jump within the program (conditional and unconditional jumps),
- synchronize with other control loops (in multi-controller setups),
- etc.

In a program the user can define his own variables and functions to do calculations, loop counting, etc. Such variables and functions are useful to perform conditional jumps within the program.
A sequencer program can only be downloaded when in sequencer 'setup' mode. Loading a new pro- gram immediately destroys an existing program even if program checking detects program errors. A change into 'program' mode and hence starting a program is only possible after successfully down- loading a program. Otherwise the state change will result in an error message.

7. Control loop

The control basically consists of the data acquisition (see chapter 9), the synthesizer (see chapter 8.1) and the control algorithms (see chapter 8.2). Synthesizer and controller are strongly related, due to synchronized operations.

7.1. Synthesizer

Architecture

Der Synthesizer besteht aus mehreren Funktionsgeneratoren (siehe Figur 7), welche Sollwerte für den Regler und Dithersignale für das Fahrsignal erzeugen. Die drei Generatoren für den Regler unterstehen einer gemeinsamen Verwaltung. Sie werden jeweils synchron gestartet, arbeiten dann aber weitgehend autonom.

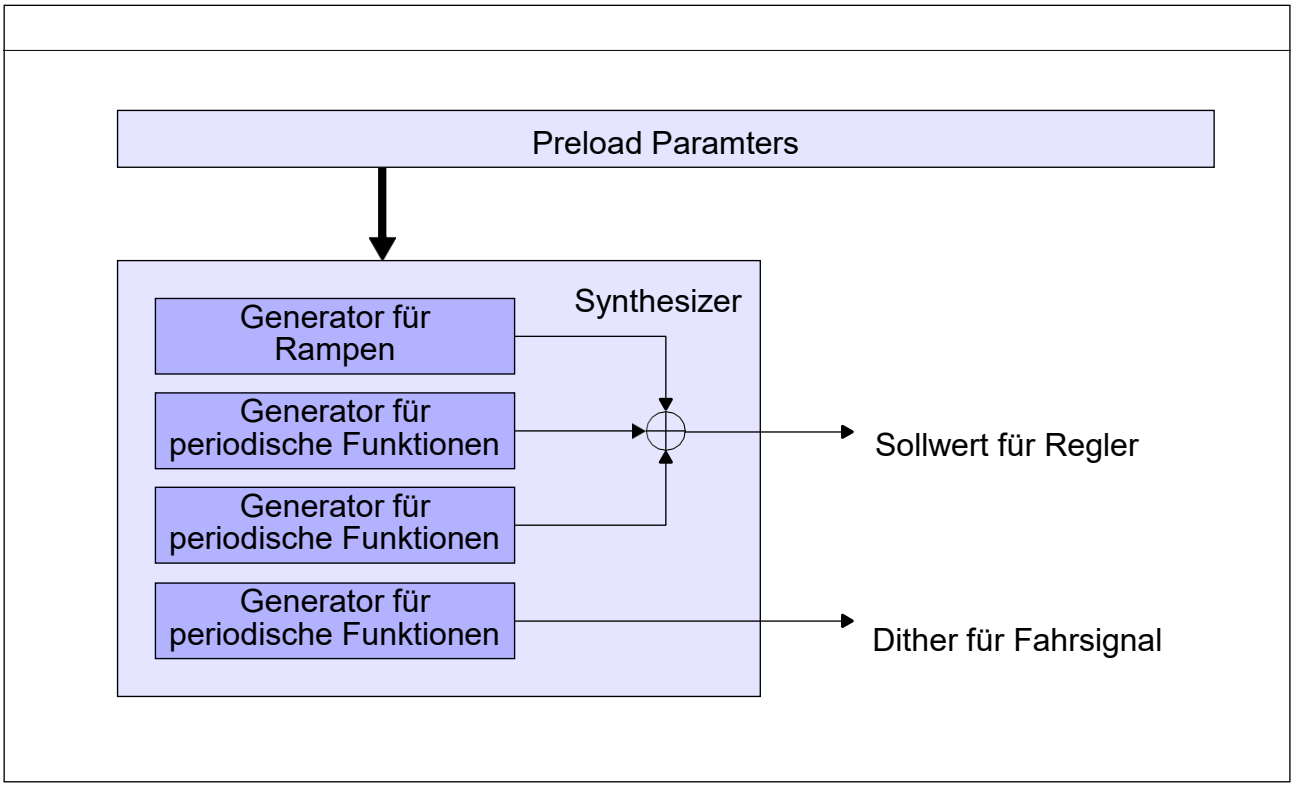


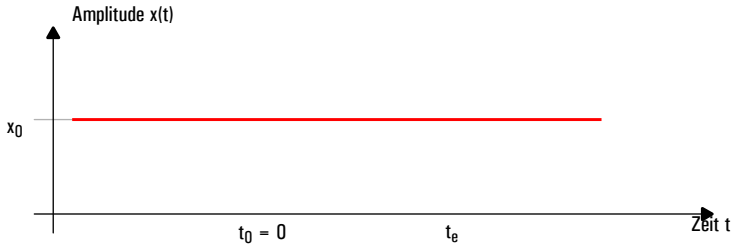
Figure 7: Schematische Darstellung des Synthesizers.

Von den Generatoren ist der Rampengenerator zuständig für absolute Grössen, die mit dem gewählten Regel- resp. Messkanal korrespondieren. Beim Wählen eines Messkanals für die Rege- lung übernimmt der Rampengenerator den Momentanwert des betreffenden Messkanals als Startw- ert. Die Generatoren für periodische Funktionen arbeiten dagegen relativ. Ihr Start- und Endwart ist immer null.
Die Sollwertgeneratoren sind mit Endkriterien ausgerüstet, die definieren, wann eine Sollwertkurve ihr Ende erreicht hat. Ein Kriterium bezieht sich auf die Kurvendauer in Sekunden, ein anderes bez- ieht sich auf periodische Funktionen und zählt die Anzahl Zyklen. Jedes Endkriterien ist entweder gesetzt oder nicht gesetzt. Ist keines der Endkriterien gesetzt, so läuft die betreffende Sollwertkurve bis zu einem manuellen Abbruch, ansonsten bis zum Erreichen des ersten Endkriteriums.
Alle Generatoren haben einen Reset-Zustand, der als Betriebsmodus gewählt werden kann. In dieser Betriebsart werden keine Abbruchkriterien überprüft. Der Ausgangswert des Rampengenerators bleibt konstant, die Generatoren für periodische Funktionen liefern null.

Der Synthesizer hat einen Befehlseingang (siehe chapter 10.17). Auf diesem Weg erhält er Konfigu- rationen sowie Start- und Stopbefehle. Der manuelle dynamische Betrieb der Maschine (siehe chap- ter 7.1) erlaubt es, Parameter via Handrad der Fernbedienung einzustellen. Die entsprechenden Befe- hle treffen ebenfalls auf diesem Weg beim Synthesizer ein.

Waveforms

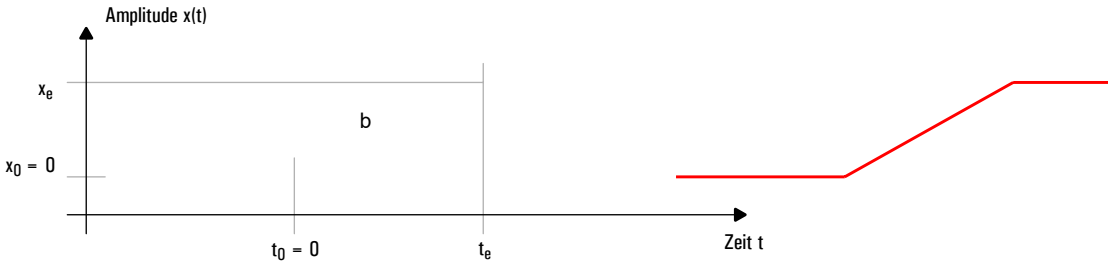
Konstant (Rampengenerator)



Die konstante Funktion erzeugt einen immer gleichen Werte x_0. Beim Start des Rampengenerators aus dem Reset-Zustand entspricht der Wert x_0 dem Anfangswert des betreffenden Messkanals. Unter gewissen Bedingungen lässt sich der Wert zum Beispiel mit dem Handrad der Fernbedienung verändern. Die Konstante Funktion kennt nur einen Parameter:

- Endzeit $t_e > 0$ in s.

Rampe (Rampengenerator)



Eine Rampe startet immer bei der aktuellen Amplitude x_0 zum Zeitpunkt $t_0 = 0$. Die Rampe ist char- akterisiert durch die folgenden Parameter:

- Steigung $b_1 > 0$, in Amplitudeneinheiten pro Sekunde.
- Steigung $b_2 > 0$, in Amplitudeneinheiten pro Sekunde (Endwert nach Beschleunigung).
- Beschleunigung $a > 0$, in Amplitudeneinheiten pro Sekundenquadrat.
- Absoluter Endwert x_e in Amplitudeneinheiten.
- Endzeit $t_e > 0$ in s.

Die Steigung b soll immer als Betrag angegeben werden. Die Steuerung entscheidet auf Grund von Start- und Endwert ob die Rampe aufwärts oder abwärts laufen muss. Die Beschleunigung a soll ebenfalls als Betrag angegeben werden. Sie darf auch Null sein. In diesem Fall wird nicht beschleu- nigt, sondern mit der Steigung b_1 gefahren, andernfalls wird linear beschleunigt von b_1 nach b_2 .

Auf Grund der beiden Parameter b und x_e ergibt sich der Zeitpunkt t_e am Ende der Rampe gemäss (gilt nur falls a = 0):

$$t_e = \frac{x_e}{b} \qquad \text{Eq. 1}$$

Gesamthaft kann die Rampe durch die folgenden Parameterkombinationen definiert werden:

- Steigung b1, absoluter Endwert xe.
- Steigung b1, Dauer der Rampe te.
- Steigungen b1 und b2 , Beschleunigung a, Endwert oder Dauer.

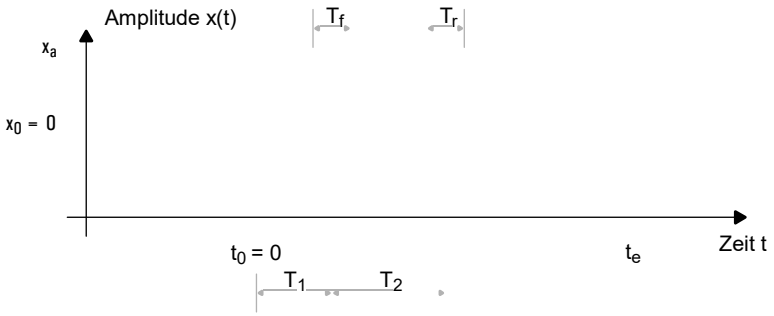
Der Rampengenerator lässt sich wie folgt charakterisieren:

- Zustände: RESET, CONST, RAMP
- In den Zuständen RESET und CONST wird beim Start der aktuelle Messwert eingelesen.
- Nach dem Ende einer Rampe wird der Zustand CONST angenommen und die aktuelle Steigung gespeichert.
- Die Vorzeichen von Steigung und Beschleunigung werden aus der aktuellen Position und der Endposition ermittelt. Sie bleiben während der Bewegung gleich.
- Wird die Beschleunigung mit 0 angegeben, wird nicht beschleunigt und immer mit Steigung = b1 gestartet (rückwärts kompatibler Modus).
- Die Variable RAMP_ACTUAL_RATE (vergl. Table 20) enthält die momentane Steigung, resp. die letzte verwendete Steigung einer beendeten Rampe. Damit lassen sich beschle- unigte Rampen mit gleicher Steigung aneinander fügen.
- Sofern bei einer beschleunigten Rampe der Endwert der Steigung 0 ist und dies erreicht wird, so endet die Rampe hier, auch wenn die Endposition und die Endzeit noch nicht erreicht sind.

Parameter:

type:	reset | const | ramp | acc
rate:	> = 0 (slope at start of acceleration)
end_rate:	> = 0 (slope at end of acceleration)
acc:	> = 0 (0: no acceleration)
stream:	primary measurement
stream stream2:	seconary measurement stream
end_ampl:	end value
end_time:	time limit in seconds (0: endless)

Trapez (Generator für periodische Funktionen)



Die Trapezfunktion startet immer bei der aktuellen Amplitude $x_0 = 0$ zum Zeitpunkt $t_0 = 0$. Am Ende t_e kehrt sie zum Amplitudenwert x_0 zurück. Die Funktion ist charakterisiert durch die folgenden Parameter:

- Amplitude x_a, negativ oder positiv, in Amplitudeneinheiten.
- Frequenz $f = T^{-1} = (T_1 + T_2)^{-1} > 0$ in Hz.
- Funktionsdauer $t_e \geq 0$ in s.
- Tastverhältnis:

$$\tau = \frac{T_1}{T} \quad 0 < \tau < 1 \qquad \text{Eq. 2}$$

- Anstiegszeit:

$$\xi = \frac{T_r}{T} \quad 0 \leq \xi < [2 \cdot \min(\tau, 1 - \tau) - \zeta] \qquad \text{Eq. 3}$$

- Abfallszeit:

$$\zeta = \frac{T_f}{T} \quad 0 \leq \zeta < [2 \cdot \min(\tau, 1 - \tau) - \xi] \qquad \text{Eq. 4}$$

- Phase $\varphi \geq 0$ in Grad[5].

Ist die Amplitude negativ, so startet die Trapezfunktion mit einer negativen Flanke, sonst mit einer positiven (wie abgebildet). Mit Hilfe des Tastverhältnisses kann die erste Phase des Trapezes gegenüber der zweiten verkürzt oder verlängert werden. Auf Grund der Funktionsdauer und der Fre- quenz lässt sich angeben, aus wie vielen Trapezperioden N die Funktion besteht:

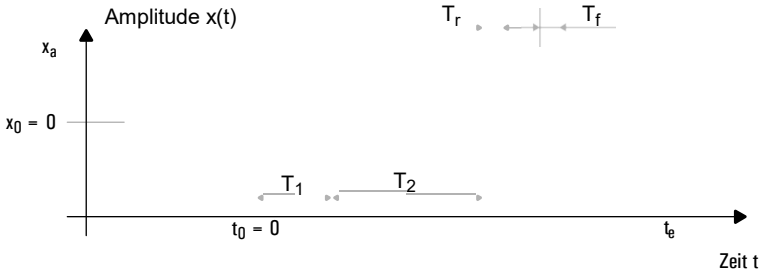
$$N = \frac{t_e}{T} \qquad \text{Eq. 5}$$

5. Die Bedingung, dass die Phase nicht kleiner als null sein kann leitet sich aus der Kausalität ab. Eine Phase grösser als null führt zu einer Verzögerung des Signal.

Folglich kann die Trapezfunktion durch folgende Parameterkombinationen definiert werden:

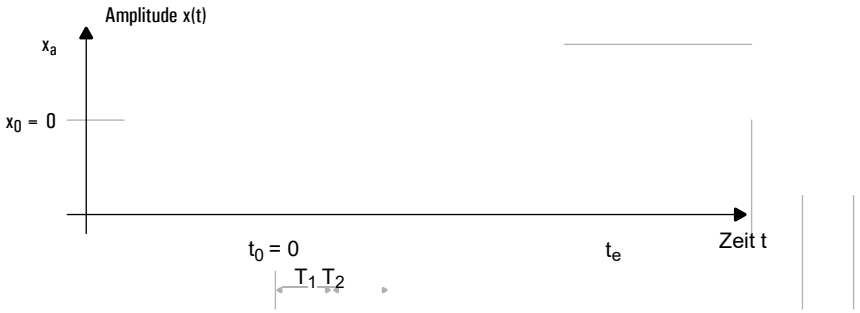
- x_a, f, t_e, τ, ξ, ζ.
- x_a, f, N, τ, ξ, ζ.
- x_a, N, t_e, τ, ξ, ζ.

Trapezpuls (Generator für periodische Funktionen)



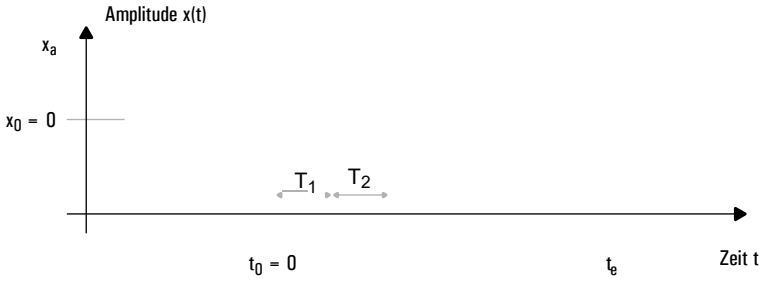
Die Trapezpulsfunktion ist identisch mit der Trapezfunktion, ausser dass die Funktion in der Phase T_2 null bleibt. Die Trapezpulsfunktion startet immer bei der aktuellen Amplitude $x_0 = 0$ zum Zeit- punkt $t_0 = 0$. Am Ende t_e kehrt sie zum Amplitudenwert x_0 zurück. Die Funktion ist durch die selben Parameter charakterisiert wie die Trapezfunktion.

Rechteck (Generator für periodische Funktionen)



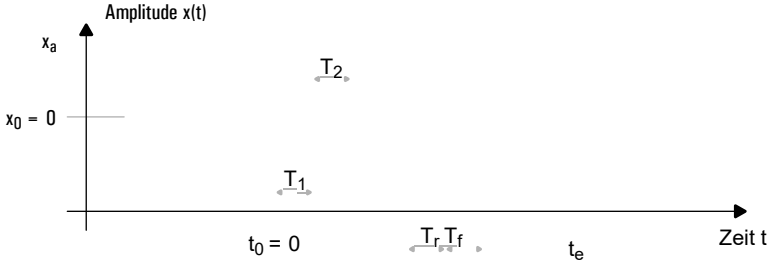
Die Rechteckfunktion ist ein Spezialfall der Trapezfunktion, bei der die Anstiegszeit T_r und die Abfallszeit T_f null sind. Die Charakterisierung ist daher identisch wie bei der Trapezfunktion, mit
$\xi = \zeta = 0$.

Rechteckpuls (Generator für periodische Funktionen)



Die Rechteckpulsfunktion ist abgeleitet aus der Trapezfunktion, resp. aus der Rechteckfunktion, wobei die Amplitude in der Phase T_2 null bleibt.

Dreieck (Generator für periodische Funktionen)



Eine Dreieckfunktion startet immer bei der aktuellen Amplitude $x_0 = 0$ zum Zeitpunkt $t_0 = 0$. Am Ende t_e kehrt sie zum Amplitudenwert x_0 zurück. Die Funktion ist charakterisiert durch die fol- genden Parameter:

- Amplitude x_a, negativ oder positiv, in Amplitudeneinheiten.
- Frequenz $f = T^{-1} = (T_r - T_f)^{-1} > 0$ in Hz.
- Funktionsdauer $t_e \geq 0$ in s.
- Tastverhältnis:

$$= \frac{T_r}{T} \quad 0 \leq \tau \leq 1 \qquad \textbf{Eq. 6}$$

- Phase $\varphi \geq 0$ in Grad[6].

Ist die Amplitude negativ, so startet die Rechteckfunktion mit einer negativen Flanke, sonst mit einer positiven (wie abgebildet). Mit Hilfe des Tastverhältnisses kann der Dreieck verformt werden, im Extremfall bis zu einem vorwärts oder rückwärts laufenden Sägezahn. Auf Grund der Funktions- dauer und der Frequenz lässt sich angeben, aus wie vielen Dreieckperioden N die Funktion besteht:

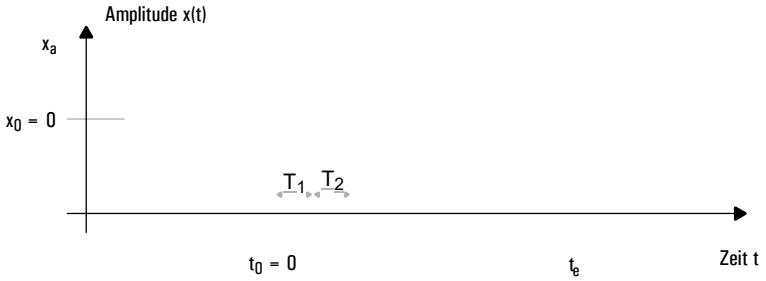
$$N = \frac{t_e}{T} \qquad \textbf{Eq. 7}$$

6. Die Bedingung, dass die Phase nicht kleiner als null sein kann leitet sich aus der Kausalität ab. Eine Phase grösser als null führt zu einer Verzögerung des Signal.

Folglich kann die Dreieckfunktion durch folgende Parametersätze definiert werden:

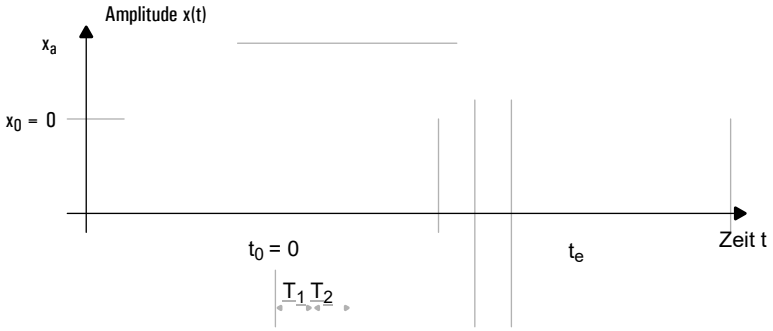
- x_a, f, t_e, τ, φ.
- x_a, f, N, τ, φ.
- x_a, N, t_e, τ, φ.

Dreieckpuls (Generator für periodische Funktionen)



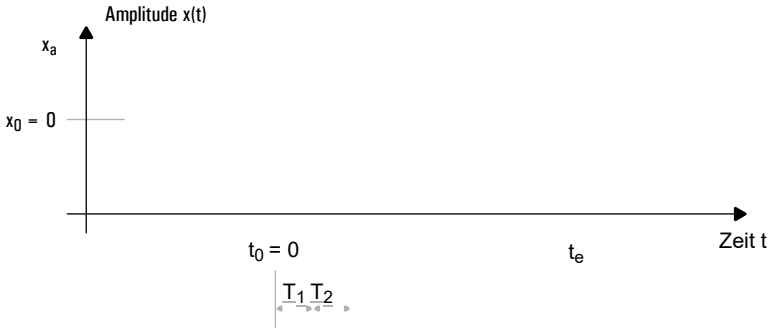
Die dargestellte Dreieckpulsfunktion entspricht der Dreieckfunktion, wobei die Amplitude in der Phase T_2 null bleibt. Zu beachten ist, dass bei der Dreieckfunktion auf Grund der Symmetrie die Phasen T_1 und T_2 immer gleich lang sind.
Die Spezifikation der Dreieckpulsfunktion entspricht jener der Dreieckfunktion.

Sägezahn 1 (Generator für periodische Funktionen)



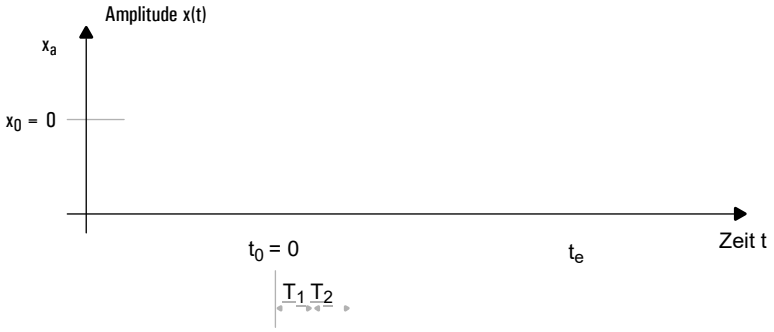
Die Sägezahnfunktion 1 ist ein Spezialfall der Dreieckfunktion, bei der $T_r = 0$ ist und entsprechend $T_f = 1$.
Das Tastverhältnis ist $\tau = 0$. Im Übrigen entspricht die Spezifikation der Dreieckfunktion.

Sägezahnpuls 1 (Generator für periodische Funktionen)



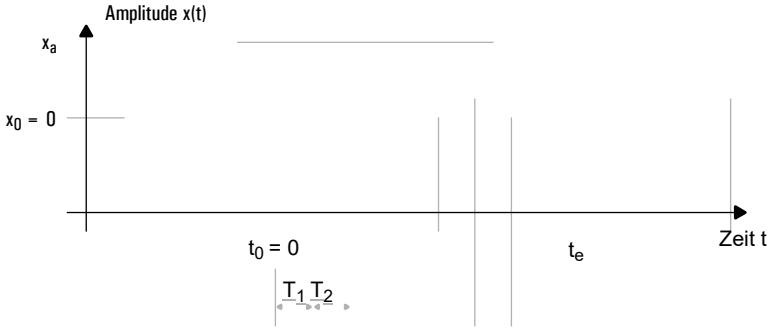
Die Sägezahnpulsfunktion 1 ist vom Sägezahn 1 abgeleitet, wobei die Amplitude während der Phase T_2 auf null bleibt. Die Spezifikation entspricht im übrigen jener der Dreieckfunktion, resp. der Sägezahnfunktion 1.

Sägezahn 2 (Generator für periodische Funktionen)



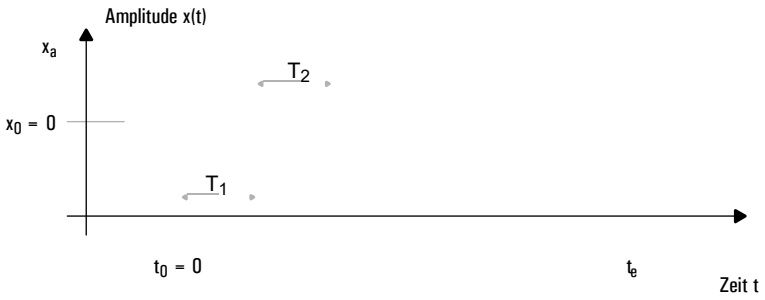
Die Sägezahnfunktion 2 ist ein Spezialfall der Dreieckfunktion, bei der $T_r = 1$ ist und $T_f = 0$. Entsprechend ist das Tastverhältnis $\tau = 1$. Im Übrigen entspricht die Spezifikation der Dreieckfunk- tion.

Sägezahnpuls 2 (Generator für periodische Funktionen)



Die Sägezahnpulsfunktion 2 ist vom Sägezahn 2 abgeleitet, wobei die Amplitude während der Phase T_2 auf null bleibt. Die Spezifikation entspricht im übrigen jener der Dreieckfunktion, resp. der Sägezahnfunktion 2.

Sinus (Generator für periodische Funktionen)



Eine Sinusfunktion startet immer bei der aktuellen Amplitude $x_0 = 0$ zum Zeitpunkt $t_0 = 0$. Am Ende t_e kehrt sie zum Amplitudenwert x_0 zurück. Die Funktion ist charakterisiert durch die folgenden Parameter:

- Amplitude x_a, negativ oder positiv, in Amplitudeneinheiten.
- Frequenz $f = T^{-1} > 0$ in Hz.
- Funktionsdauer $t_e \geq 0$ in s.
- Phase $\varphi \geq 0$ in Grad[7].

Ist die Amplitude negativ, so startet die Sinusfunktion mit einer negativen Flanke, sonst mit einer positiven (wie abgebildet). Auf Grund der Funktionsdauer und der Frequenz lässt sich angeben, aus wie vielen Sinusperioden N die Funktion besteht:

$$N = \frac{t_e}{T} \qquad \text{Eq. 8}$$

Folglich kann die Sinusfunktion durch folgende Parametersätze definiert werden:

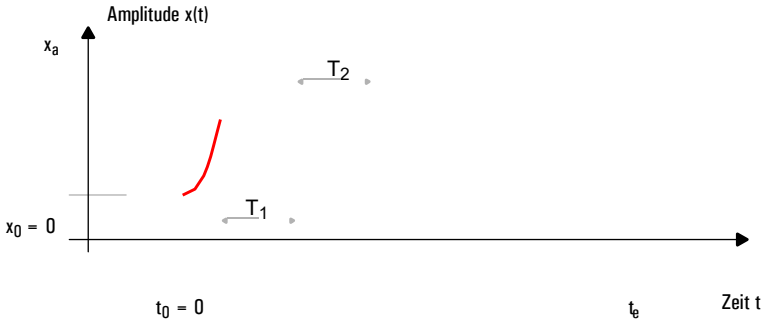
- x_a, f, t_e, φ.
- x_a, f, N, φ.
- x_a, N, t_e, φ.

Sinuspuls (Generator für periodische Funktionen)

Die Sinuspulsfunktion ist vom Sinus abgeleitet, wobei die Amplitude während der Phase T_2 auf null bleibt. Die Spezifikation entspricht im Übrigen der Sinusfunktion.

7. Die Bedingung, dass die Phase nicht kleiner als null sein kann leitet sich aus der Kausalität ab. Eine Phase grösser als null führt zu einer Verzögerung des Signal.führt zu einer Verzögerung des Signal.

Haversinus (Generator für periodische Funktionen



Der Haversinus ist verwandt mit der Sinusfunktion, ist aber einseitig bezüglich x_0. Er startet immer bei der aktuellen Amplitude $x_0 = 0$ zum Zeitpunkt $t_0 = 0$. Am Ende t_e kehrt er zum Amplitudenwert x_0 zurück. Die Funktion ist charakterisiert durch die folgenden Parameter:

- Amplitude x_a, negativ oder positiv, in Amplitudeneinheiten.
- Frequenz $f = T^{-1} > 0$ in Hz.
- Funktionsdauer $t_e \geq 0$ in s.
- Phase $\varphi \geq 0$ in Grad[8].

Ist die Amplitude x_a negativ, so startet die Haversinusfunktion mit einer negativen Flanke, sonst mit einer positiven (wie abgebildet). Auf Grund der Funktionsdauer und der Frequenz lässt sich angeben, aus wie vielen Sinusperioden N die Funktion besteht:

$$N = \frac{t_e}{T} \qquad \text{Eq. 9}$$

Folglich kann die Haversinusfunktion durch folgende Parametersätze definiert werden:

- x_a, f, t_e, φ.
- x_a, f, N, φ.
- x_a, N, t_e, φ.

Sweep

Die Parameter Amplitude und Frequenz der Wavegeneratoren lassen sich durch bestimmte Funk- tionen auch automatisch verändern. Mit der Sweepfunktion ändern sich diese automatisch mit einer bestimmten Geschwindigkeit von einem Start zu einem Endwert. Nach Erreichen des Endwertes wird dieser bis auf weiteres beibehalten. Die Aenderung kann sowohl linear als auch exponentiell erfolgen. Für die Amplitude A(t) gilt bei linearer Aenderung:

$$A(t) = A_{start} + k_l \cdot (t - t_0) \qquad \text{Eq. 10}$$

Bei exponentieller Aenderung gilt:

$$A(t) = A_{start} \; k_e^{(t - t_0)} \qquad \text{Eq. 11}$$

- A_{start} ist der Startwert der Amplitude.
- t_0 ist der Startzeitpunkt.
- k_l ist die lineare Steigung.
- k_e ist die exponentielle Steigung.

Folgendes gilt grundsätzlich für die Sweepfunktion:

- Die Sweepfunktion wird nicht weiter ausgeführt, sobald A(t) den Endwert erreicht hat.
- Sie ist auf alle Wave-Typen anwendbar.
- Für die Frequenz gilt dies alles synonym.
- Der von der Sweepfunktion gesteuerte Parameter ist nicht trimmbar.
- Die lineare Steigung k_l kann grundsätzlich positiv oder negativ sein. Allerdings wird das richtige Vorzeichen automatisch aus Start- und Endwert berechnet.
- Die exponentielle Steigung k_e muss immer grösser als Null sein.
- Bei einem exponentiellen Sweep muss der Startwert grösser als Null sein. Falls Null angegeben wird, so wird ein Defaultwert von 10^{-3} verwendet (Hz oder entsprechende Einheit).
- Ist die Endamplitude negativ, so wird dies automatisch auch für die Startamplitude angenommen.

Folgende Parameter werden im Zusammenhang mit Sweep benötigt:

start_ampl, start_freq:	Startwerte
ampl, freq:	Endwerte
amplswp_type, freqswp_type:	“none” | ”lin” | ”log”
amplswp_rate, freqswp_rate:	Steigungen

Modulation

Die Frequenz, resp. die Amplitude eines Wavegenerators kann auch durch das Ausgangssignal eines anderen Generators moduliert werden. Damit können z.B. Signale generiert werden, deren Ampli- tude oder Frequenz sich zyklisch ändert. Als Quelle für das Modulationssignal kommen sowohl Wave-Generatoren als auch ein externer Stream oder der Curve-Generator (Wavefile) in Frage.

Für die Amplitudenmodulation gilt folgendes:

$$A(t) = A_0 + s(t) \qquad \text{Eq. 12}$$

- A(t) ist die momentane Amplitude des modulierten Generators.
- A_0 ist definierte Amplitude gemäss Parameter.

- s(t) ist das Ausgangssignal der Modulationsquelle (Wave, Stream oder Curve).

Für die Frequenzmodulation gilt folgendes:

$$f(t) = f_0 + s(t) \qquad \text{Eq. 13}$$

- f(t) ist die momentane Frequenz des modulierten Generators.
- f_0 ist die definierte Frequenz gemäss Parameter.
- s(t) ist das Ausgangssignal der Modulationsquelle (Wave, Stream oder Curve).

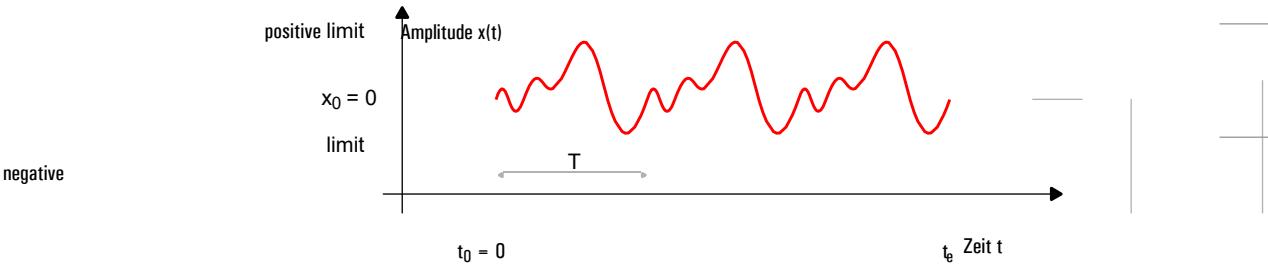
Bei der Modulation gilt ausserdem folgendes:

- Ein Wave-Generator kann sich nicht selbst modulieren.
- Zwei Wave-Generatoren können sich nicht gegenseitig modulieren.
- Das Signal eines Generators, der als Modulationsquelle dient, erscheint nicht mehr am Ausgang des Synthesizers.
- Erreicht ein modulierter Generator ein Endkriterium oder wird er gestoppt, so wird auch die entsprechende Modulationsquelle gestoppt.
- Ein modulierter Parameter kann nicht getrimmt werden.

Folgende Parameter werden im Zusammenhang mit Modulation benötigt:
ampl, freq: Unmodulierte Werte amplswp_type, freqswp_type:
"none" | "mod_wave1" | "mod_wave2" | "mod_wave3" | "mod_stream" | "mod_curve"

Noise
Generation of evenly distributed DC-free white noise. The noise generator only accepts the ampli- tude x_a and the end-time t_e parameter. The generated peak values are at most 50% of the given ampli- tude. The RMS amplitude is approx. 30% of the given amplitude. This generator doesn't provide cycles and therefor doesn't increment the cycle counter.

Curve (play PCM data)



This generator type plays presampled curves sample by sample. If the number of periods is larger than 1 it will start again with the first sample after the last sample. If the number of periods is set to

zero, the generator will repeat endlessly. The curve data must be provided as a wave-file which must be locally available on the controller. The curve can be influenced by five parameters:

- Gain x_a, positive or negative.
- Number of periods N.
- Positive and negative limit.
- End time

The gain directly applies to the curve data and determines its amplitude. The desired data values can never go beyond the positive limits and never below the negative limit. If the end time is unequal to 0, the playing immediately stops after the given time.
The play time t_e is given by the following equation:

$$t_e = N \cdot T = N \cdot (T_s \cdot N_k) \qquad \text{Eq. 14}$$

With $T_s = f_s^{-1}$ as the sample interval and f_s the sample frequency. N_k is the number of samples the curve consists of, and T is the period of the curve.

Supported parameters:

type:	reset | run
plimit:	upper limit (output value will never go beyond)
nlimit:	lower limit (output value will never go below)
gain:	linear gain factor
end_time:	time limit in seconds (0: endless)
end_cycles:	number of repetitions (0: endlessrepetition)
file:	name of the wave-file

Supported data formats:
WAVE type1:PCM integer data (8Bit unsigned, 16Bit signed, 32Bit signed), 1 or 2 channels.
WAVE type3:PCM float data (32Bit IEEE float), 1 or 2 channels.

Remark: When using 2-channel data channel 2 is omitted.
Wave-file transport:

To be able to play samples from a wave-file, this file must be previously copied to the local storage area of the controller. On the controller the files are stored non-volatile, therefor a certain wave-file must only be copied once. During operation, up to 10 opened wave-files are cached, to speed up con- secutive using. The transport from the host-PC to the controller is done using the well-known FTP protocol. For that purpose the controller implements a FTP-server with the following access parame- ters:

Username: pcs8kwave Password: 8kWApCveS
Wave-files are stored on a separate partition with a size of 1...3GB. The user is responsible to manage this space in a reasonable manner. A full partition will not influence any functions of the controller.

When using curve play in sequencer programs, be aware that wave-files must be locally available at the time the sequencer program is loaded. A sequencer program can use up to 10 different wave-files.

External desired signal
With this generator type, the synthesizer uses external stream data as the desired signal. Every avail- able stream except the synthesizer itself and the control stream can be used. The behaviour can be influenced by four parameters:

- Gain, positive or negative.
- Positive and negative limit.
- End time

When starting this generator the synthesizer will keep the first sample and will use it as an offset value to prevent the output from an immediate step.

Supported parameters:

type:	reset | run
plimit:	upper limit (output value will never go beyond)
nlimit:	lower limit (output value will never go below)
desired_stream:	input stream data (see Table 2)
end_time:	time limit in seconds (0: endless)

Trimming
Some waveform parameters like amplitude, frequency, etc. are trimmable during operation, for example by the hand wheel on the remote control or by buttons on the host PC.

Summary
Table 10 lists all possible synthesizer waveforms types with their corresponding names.

Type of Waveform	Waveform Type
No waveform	reset
Constant value	const
Ramp function	ramp
Accelerated ramp function	acc
Trapezoid function	trapez
Rectangular function	rect
Triangular function	triang
Backward sawtooth function	saw1

Table 10: Synthesizer waveforms.

Type of Waveform	Waveform Type
Forward sawtooth function	saw2
Sinewave function	sine
Haversine function	haversine
Noise function	noise
Curve play / external desired signal	run

Table 10: Synthesizer waveforms.

Trimmable waveform parameters are listed in table .

Trimmable Parameter	Parameter name	Used in Generator
No parameter	none	ramp, wave
Constant amplitude	const	ramp
Ramp rate	rate	ramp
Frequency	freq	wave
Amplitude	ampl	wave
Phase	phase	wave

Table 11: Trimmable parameters and parameter names.

Completion Codes

Sobald der Synthesizer bei einer laufenden Funktion ein Endkriterium detektiert, wird die Funktion beendet und dabei ein "completion code" ausgegeben. Dieser Code wird einerseits intern verwendet und andererseits via Event (Nr. 303, siehe Table 4) als Attribut ausgegeben. Folgende Tabelle zeigt alle Codes:

Funktion	Endkriterium	Completion Code
Rampe	Zeit-Endwert erreicht	0x1
Rampe	Amplituden-Endwert erreicht	0x2
Wave 1	Zeit-Endwert erreicht	0x4
Wave 1	Anzahl Zyklen erreicht	0x8
Wave 2	Zeit-Endwert erreicht	0x10

Table 12: Synthesizer completion codes

Funktion	Endkriterium	Completion Code
Wave 2	Anzahl Zyklen erreicht	0x20
Wave 3	Zeit-Endwert erreicht	0x40
Wave 3	Anzahl Zyklen erreicht	0x80
External stream	Zeit-Endwert erreicht	0x100
Wave play	Zeit-Endwert erreicht	0x200
Wave play	Anzahl Zyklen erreicht	0x400

Table 12: Synthesizer completion codes

7.2. Control algorithms

Das Reglermodul besteht aus einer Architektur, die mehrere verschiedenartige Regler nebeneinander beherbergen kann. Auf diese Weise ist es möglich, während einer Messung zu unterschiedlichen Zeiten unterschiedliche Reglerarten einzusetzen.

PID Controller

Structure

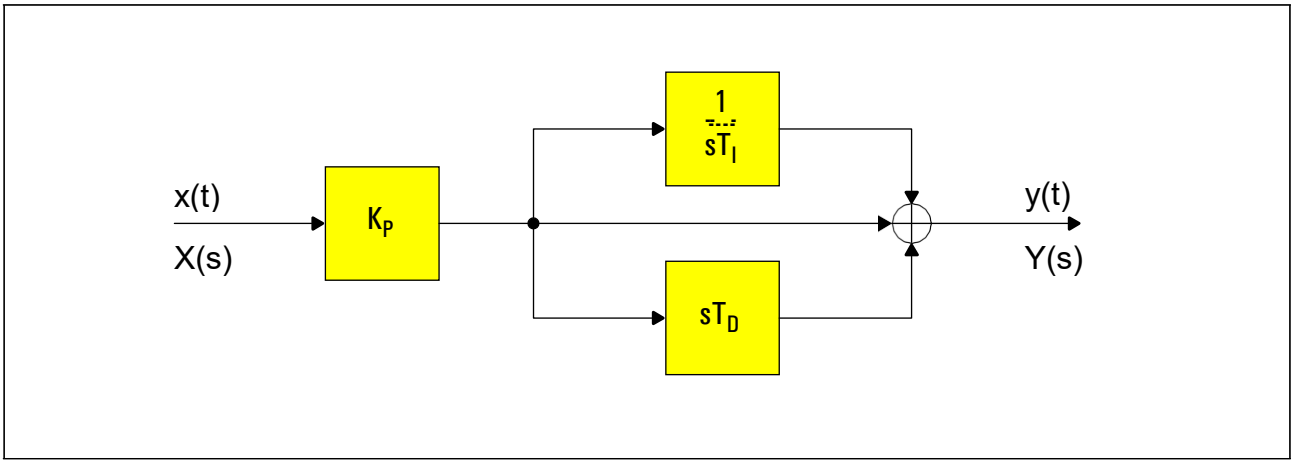


Figure 8: PID Regler in Seriestruktur.

Figur 8 zeigt den analogen PID Regler in Seriestruktur. Seine Ausgangsgleichung lautet:

$$y(t) = K_P \cdot \left\{ x(t) + \frac{1}{T_I} \int x(t) d\tau + T_D \frac{dx(t)}{dt} \right\} \qquad \text{Eq. 15}$$

Daraus folgt die Laplace-Übertragungsfunktion:

$$G(s) = \frac{Y(s)}{X(s)} = K_P \cdot \left\{ 1 + \frac{1}{sT_I} + sT_D \right\}$$ Eq. 16

Es wird nun der Integralteil Ψ_I folgendermassen durch Bilineare Transformation diskretisiert. Dabei ist T das Abtastintervall.

$$\Psi_I(s) = \frac{Y_I(s)}{X(s)} = \frac{1}{sT_I} \qquad \Psi_I(z) = \frac{1}{T_I} \cdot \frac{T(z+1)}{2(z-1)}$$

$$y_I[n] = y_I[n-1] + \frac{T}{2T_I} \cdot (x[n] + x[n-1])$$ Eq. 17

Eq. 17 wird oft auch als Trapeznäherung bezeichnet. Als Vereinfachung werden statt dessen oft die folgenden Ausdrücke für den Integralteil verwendet:

$$\tilde{y}_{I1}[n] = y_I[n-1] + \frac{T}{T_I} \cdot x[n]$$

$$\tilde{y}_{I2}[n] = y_I[n-1] + \frac{T}{T_I} \cdot x[n-1]$$ Eq. 18

Der Differentialteil Ψ_D ergibt sich mit der Bilinearen Transformation zu:

$$\Psi_D(s) = \frac{Y_D(s)}{X(s)} = sT_D \qquad \Psi_D(z) = T_D \cdot \frac{2(z-1)}{T(z+1)}$$

$$y_D[n] + y_D[n-1] = \frac{2T_D}{T} \cdot (x[n] - x[n-1])$$ Eq. 19

Wenn wir das Resultat durch 2 dividieren erhalten wir sozusagen die mittlere Ableitung zwischen den Zeitpunkten n und n+1. Dies führt zu zwei Möglichkeiten:

$$\bar{y}_{D1}[n] = \frac{y_D[n] + y_D[n-1]}{2} = \frac{T_D}{T} \cdot (x[n] - x[n-1])$$

$$\bar{y}_{D2}[n] = \frac{y_D[n+1] + y_D[n]}{2} = \frac{T_D}{T} \cdot (x[n+1] - x[n])$$ Eq. 20

Da der Differentialanteil die Eigenschaft hat, Rauschen mit zunehmender Frequenz mehr zu ver- stärken, wird oft ein Mittelwert aus mehreren aufeinanderfolgenden Differentialanteilen gebildet. Im Fall der Mittelung von zwei Anteilen erhält man:

$$\begin{aligned}\hat{y}_D[n] &= \frac{1}{2}\cdot\frac{T_D}{T}\{(x[n]-x[n-1])+(x[n-1]-x[n-2])\}\\ &= \frac{T_D}{2T}\cdot(x[n]-x[n-2])\end{aligned}$$ **Eq. 21**

Damit folgt nun der diskrete PID Regler mit Integration gemäss Trapeznäherung und gemitteltem Differentialanteil:

$$y[n] = K_P\cdot\{x[n]+y_I[n]+\hat{y}_D[n]\}$$ **Eq. 22**

Eq. 22 wird oft als Positionsalgorithmus bezeichnet. Der sog. Geschwindigkeitsalgorithmus ergibt sich wie folgt:

$$\begin{aligned}\Delta y[n] &= y[n]-y[n-1]\\ &= K_P\cdot\{x[n]+y_I[n]+\hat{y}_D[n]\}-K_P\cdot\{x[n-1]+y_I[n-1]+\hat{y}_D[n-1]\}\\ &= K_P\cdot\{x[n]-x[n-1]+y_I[n]-y_I[n-1]+\hat{y}_D[n]-\hat{y}_D[n-1]\}\end{aligned}$$ **Eq. 23**

Dabei ergibt sich unter Verwendung von Eq. 17:

$$y_I[n]-y_I[n-1] = \frac{T}{2T_I}\cdot(x[n]+x[n-1])$$ **Eq. 24**

Folglich lautet der Geschwindigkeitsalgorithmus:

$$\begin{aligned}\Delta y[n] = K_P\cdot\Bigg\{&x[n]-x[n-1]+\frac{T}{2T_I}\cdot(x[n]+x[n-1])+\\ &\frac{T_D}{2T}\cdot(x[n]-x[n-1]-x[n-2]+x[n-3])\Bigg\}\end{aligned}$$ **Eq. 25**

Oder nach Eingangsvariablen sortiert:

$$\begin{aligned}\Delta y[n] = K_P\cdot\Bigg\{&x[n]\left(\frac{T}{2T_I}+\frac{T_D}{2T}+1\right)+x[n-1]\left(\frac{T}{2T_I}-\frac{T_D}{2T}-1\right)+\\ &x[n-2]\left(-\frac{T_D}{2T}\right)+x[n-3]\left(\frac{T_D}{2T}\right)\Bigg\}\end{aligned}$$ **Eq. 26**

Parameter Trimming
Table 13 lists the trimmable PID parameters.

Trimmable PID Parameter	Parameter Name	Possible Range
Controller gain	kp	0 □ kp
Integral part	ti	0 □ ti
Differential part	td	0 □ td

Table 13: Trimmable PID parameters and parameter names.

PIDT Controller with Anti-Windup
The PIDT controller is similar to the PID controller shown in figure 8 but the PIDT controller includes two extensions:

1 The D component has a limited gain, hence it is called DT component.
2 The I component is limited by an anti-windup mechanism.

DT Component
Instead of equation 16 we use the following Laplace transfer function for the controller:

$$G(s) = \frac{Y(s)}{X(s)} = K_P \cdot \left\{ 1 + \frac{1}{sT_I} + \frac{sT_D}{1 + sT_1} \right\}$$ Eq. 27

The D component from equation 16 has a gain that increases with frequency. This is not always use- ful because high frequency noise in the controller's feedback leads to dramatic jumps at the control- ler output. The DT component limits this gain for all frequencies above $1/T_1$ rad/s. Note that for nor- mal operation the following condition applies where T is the sampling interval.

$$T_D > T_1 > T$$ Eq. 28

When transforming equation 27 into the discrete time domain we obtain the same integral part as in equation 17. The bilinear transform for the DT component leads to:

$$\Psi_D(s) = \frac{Y_D(s)}{X(s)} = \frac{sT_D}{1 + sT_1} \qquad \Psi_D(z) = T_D \cdot \frac{2(z-1)}{z(T + 2T_1) + (T - 2T_1)}$$

$$y_D[n] = \frac{2T_D}{T + 2T_1} \cdot (x[n] - x[n-1]) + y_D[n-1] \cdot \frac{2T_1 - T}{2T_1 + T}$$ Eq. 29

Note that with equation 28 the following applies:

$$\frac{1}{3} < \frac{2T_1 - T}{2T_1 + T} < 1 \qquad \text{Eq. 30}$$

Hence the output of equation 29 decays with x[n] = x[n-1]. In fact equation 29 is a first order averag- ing filter as desired.

Anti-Windup
The anti-windup mechanism considers two aspects:

1 Do only integrate when the controller's output signal does not reach its limits ζ.

2 Apply another limit ζ to the integral part.

The anti-windup solution is shown in equation 31. First the intermediate value $\hat{y}_I[n]$ equation 17. Then the intermediate value is compared to a threshold ξ

$$\hat{y}_I[n] = \begin{cases} y_I[n-1] & \text{if } y[n-1] \geq \zeta \\ y_I[n-1] + \frac{T}{2T_I} \cdot (x[n] + x[n-1]) & \text{if } y[n-1] < \zeta \end{cases}$$

$$y_I[n] = \begin{cases} -\xi & \text{if } \hat{y}_I[n] < -\xi \\ \hat{y}_I[n] & \text{if } |\hat{y}_I[n]| < \xi \\ \xi & \text{if } \hat{y}_I[n] > \xi \end{cases} \qquad \text{Eq. 31}$$

Controller Equation
The controller output equation is given in equation 32.

$$y[n] = K_P \cdot (x[n] + y_I[n] + y_D[n]) \qquad \text{Eq. 32}$$

Refer to equation 31 for $y_I[n]$ and to equation 29 for $y_D[n]$.

Parameter Trimming
Table 14 lists the trimmable PIDT parameters.

Trimmable PIDT Parameter	Parameter Name	Possible Range
Controller gain	kp	0 □ kp
Integral part	ti	0 □ ti
Differential part	td	0 □ td
First order differential part limiting(1)	t1	T □ t1 □ td

Table 14: Trimmable PIDT parameters and parameter names.

1. T denotes the sampling interval.

PIDM Mixed Stroke and Force Controller
Normally the PIDM controller is used to control stroke. But when for some reason a force accrues the controller tries to minimize the combination of force and deviation from the desired stroke. Therefor the PIDM is called a mixed stroke and force controller. Figure 9 shows its layout.

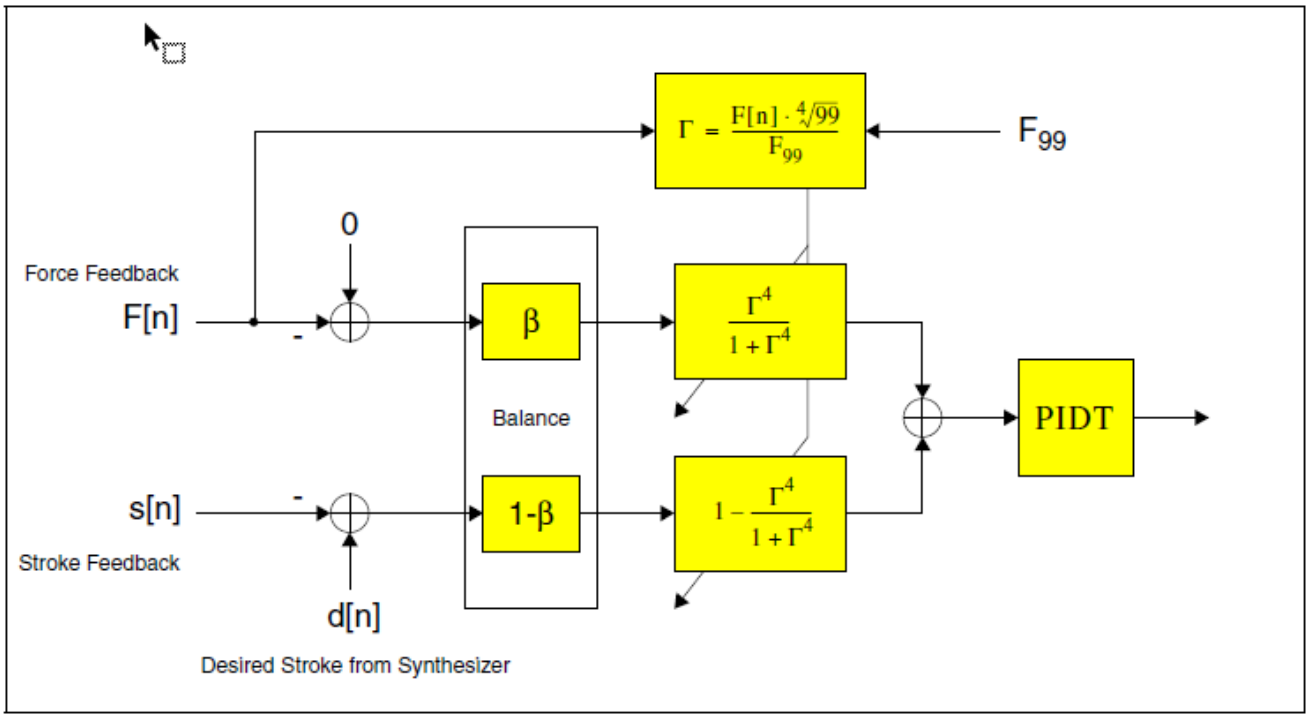


Figure 9: PIDM controller layout.

The controller has two inputs, one for measured stroke and one for measured force. First the differ- ence between desired and measured values are taken. Note that the desired force is zero. In a balance stage the stroke and force error samples are brought into the same scale. For example, stroke error is

normally measured in millimeters and force error in newton or kilonewton. Hence force is scaled between 10^3 and 10^6 times stroke. Therefor the balance factor β would normally be something like 10^{-3} to 10^{-6}.

The force samples are also used to derive a weighting factor Ω [n], see equations 33 and 34.

$$\Omega[n] = \frac{\Gamma[n]^4}{1+\Gamma[n]^4} \qquad \text{with} \qquad \Gamma[n] = F[n] \cdot \frac{\sqrt[4]{99}}{F_{99}} \qquad \text{Eq. 33}$$

$$\Omega[n] = \frac{F[n]^4}{\frac{F_{99}^4}{99} + F[n]^4} \qquad \text{Eq. 34}$$

Parameter F_{99} is a user-defined constant which says at what actual force F[n] the controller becomes a 99 % force controller. Figure 10 illustrates the effect of the weighting factor. Ω [n] is used as a weight for force, (1 - Ω [n]) is used as a weight for stroke. With raising force the weight smoothly transitions from pure stroke control to pure force control.

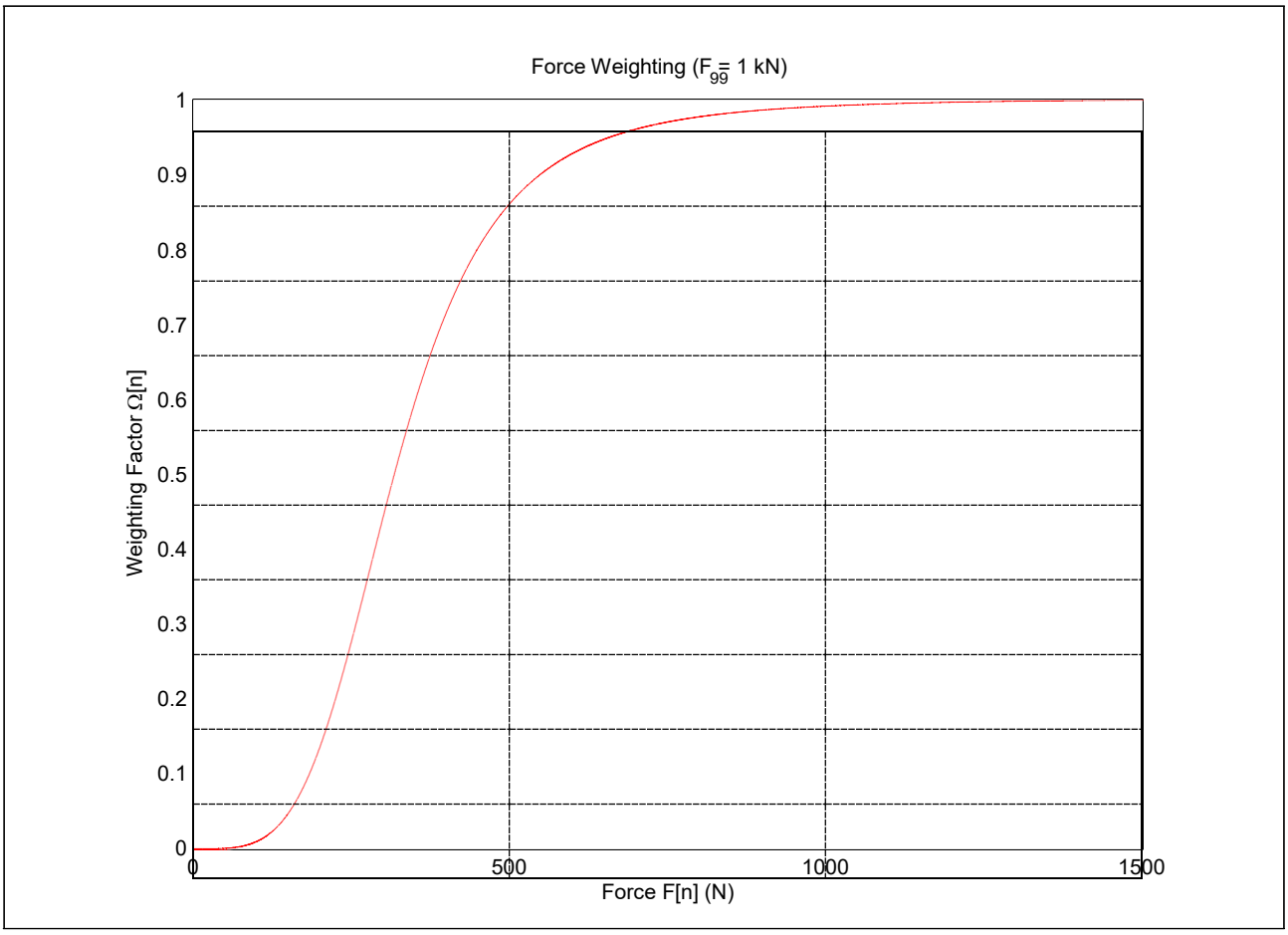


Figure 10: Weighting factor Ω [n] for F_{99} = 1 kN

Finally the combined error signal is fed into a standard PIDT controller, as described above.

Parameter Trimming
Table 15 lists the trimmable PIDM parameters.

Trimmable PIDM Parameter	Parameter Name	Possible Range
Controller gain	kp	$0 \leq kp$
Integral part	ti	$0 < ti$
Differential part	td	$0 \leq td$
First order differential part limiting(1)	t1	$T < t1 < td$
Balance	bal	$0 \leq bal \leq 1$
Weighting factor	f99	$0 \leq f\ 99$

Table 15: Trimmable PIDM parameters and parameter names.
1. T denotes the sampling interval.

PIDT_peak Adaptive peak mode Controller
Description to follow.

PIDT_peak2 Improved adaptive peak mode Controller
The controller output equation used to control the peak of a desired synthesizersignal is given in Eq. 32 on page 59 (PIDT Controller with Anti-Windup). However it is recommanded to disable the IDT- Components (T_I = 100000, T_D = 0).

To reach a given amplitude the ac-gain and dc-offset is adapted apart: The ac-gain is scaled with a factor kx_{AC} and the dc-offset is overlie with an additional offset kx_{DC}. This leads to a distortion of the reference.
By default kx_{AC} = 1 and kx_{DC} = 0 (reference without distortion). The PIDT_Pk2 Con- troller layout is given in Figur 11.

Peakdetection is done block by block. In a window approximately as long as the period of the ac-part of the syntesizersignal (windowsize) the maximum and minimum is detected to calculate the peak to peak value (ref_pp). In the same window the maximum and minimum of the actual measured posi- tion is evaluated (act_max, act_min). Filtering with a 4th order butterworthfilter avoids abrupt changes. Adaption is done by comparing the reference from the syntesizer with the measured data. The following listing should clarify this:

windowsize 1.2 * signalperiod (from synthsizer)
Peakdetection: ref_max, ref_min (Peaks of the reference) act_max, act_min (Peaks of measured data) peak to peak: ref_pp = ref_max - ref_min

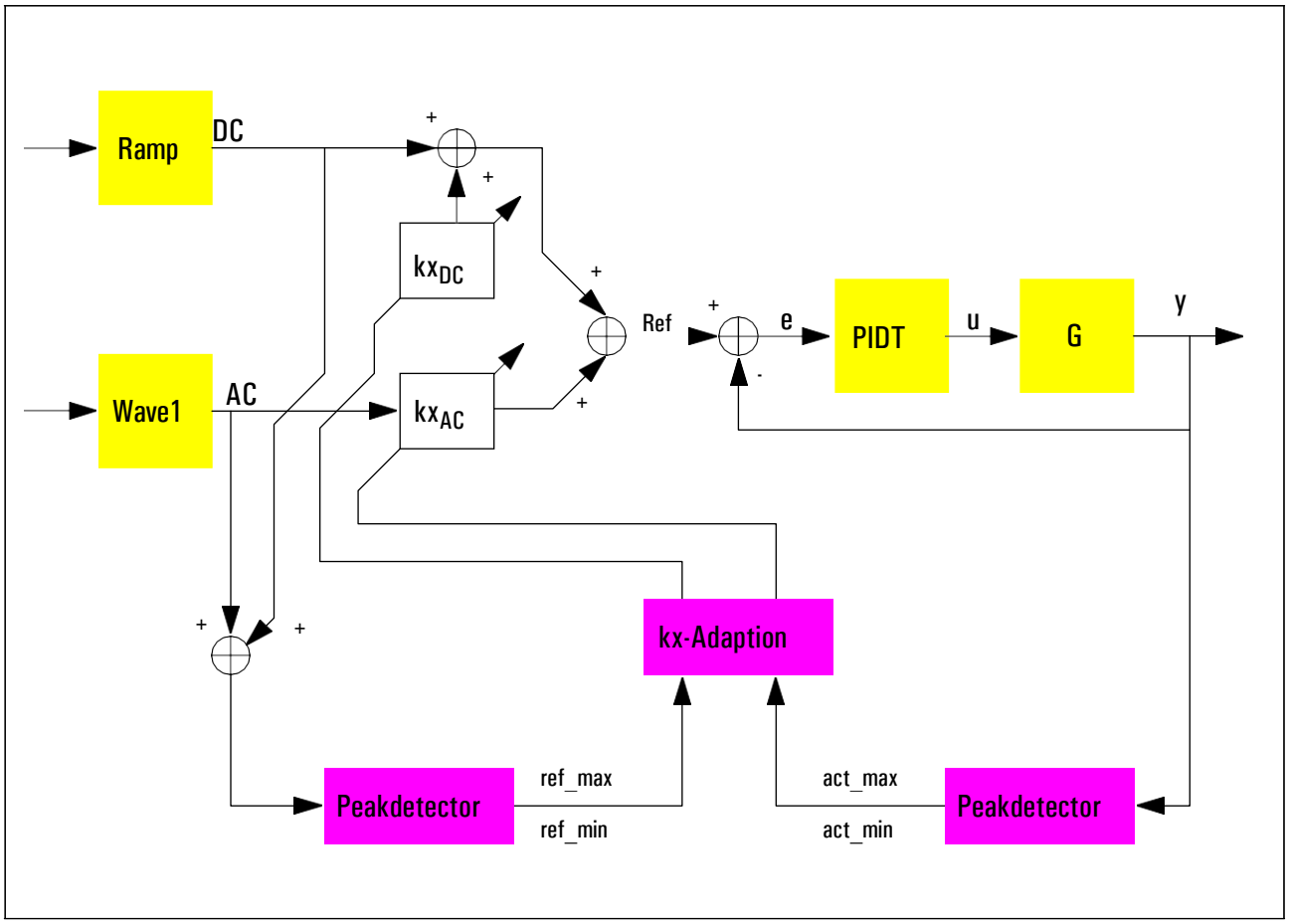


Figure 11: PIDT_Pk2 controller layout

	$act\_pp = act\_max - act\_min$
Filtering:	ref_pp_fil (butter 4, 8/4000)
	act_pp_fil (butter 4, 8/4000)
AC-Adaption:	$kx_{AC}\_nominal = ref\_pp\_fil / act\_pp\_fil$
	$kx_{AC} = kx_{AC} * kx_{AC}\_nominal^{1/(3*windowsize)}$
DC-Adaption:	$kx_{DC}\_nominal = (ref\_max + ref\_min) / 2 \ldots$
	$\ldots -(act\_max + act\_min) /2$
	$kx_{DC} = kx_{DC} + kx_{DC}\_nominal / (2 * windowsize)$
Limitation:	$kx_{AC} = [0.2 \ldots 10]$
	$kx_{DC} = [-ref\_pp \ldots ref\_pp]$
New reference:	$Ref = (AC * kx_{AC}) + (DC + kx_{DC})$

The adaption factor kx_{AC} is adapted sample wise so that it reaches the desired value kx_{AC}_nominal after approximately 3.6 periods (3*windowsize). The aditional offset kx_{DC} is adapted sample wise so that it reaches the desired value kx_{DC}_nominal after approximately 2.4 periods (2*windowsize). The postprocessing of kx_{AC}_nominal and kx_{DC}_nominal is done do avoid steps in the reference for the controller. Furthermore it is ensured that adaption converges uniformly to the desired state.

To avoid overshoot kx_{AC} and kx_{DC} are reset when a synthesizer parameter is trimmed.

Parameter Trimming
Table 16 lists the trimmable PIDT_peak2 parameters.

Trimmable PIDT_peak2 Parameter	Parameter Name	Possible Range
Controller gain	kp	$0 \leq kp$
Integral part	ti	$0 < ti$
Differential part	td	$0 \leq td$
First order differential part limiting(1)	t1	$T < t1 < td$

Table 16: Trimmable PIDT_peak2 parameters and parameter names.
1. T denotes the sampling interval.

PIDV enhanced PID Controller in velocitymode
The PIDV-Controller is a PID-Controller in velocitymode. It is used when the plant is a integrating system (e.g. DC-Motor). To deal with various nonlinearities some user specific add-ons can be acti- vated when necessary. The detailed description of the PIDV-Controller and its parameters can be found in the report "UN110826_PIDV-Reglerdoku.pdf" [2].

Parameter Trimming
Table 17 lists the trimmable PIDV parameters.

Trimmable PIDV Parameter	Parameter Name	Possible Range
Quadratic gain	ke2	$0 \leq ke2$
Feed forward gain	kff	$0 \leq kff$
Controller gain	ke	$0 \leq ke$
Static drift compensation	sdc	$-1 \leq sdc \leq 1$
DC-Correction gain	kdcc	$0 \leq kdcc$
Peakcontroller	pk	$pk = 0\ pk \geq 1$ (1)
Gain for residual error integrator	krei	$0 \leq krei$
Timeconstant for residual error integr.	tau	$0 \leq tau$
Errorlimit to deplete rei	elim	$0 \leq elim$
Manual balancer	sym_man	$0 \leq sym\_man$

Table 17: Trimmable PIDV parameters and parameter names.

Trimmable PIDV Parameter	Parameter Name	Possible Range
Automatic balancer	sym_adp	0, 0.5...2[2]
Leckage gain	kl	0 ≤ kl
Lower force	f0	0 ≤ f 0
Upper force	f1	0 ≤ f 1
Kicker gain	gki	0 ≤ gki
Kicker timeconstant	tki	0 ≤ tki

Table 17: Trimmable PIDV parameters and parameter names.

1. pk roughly denotes the number of periods which are required to reach the desired amplitude.
2. 0: disabled; 1: symmetrical behaviour

PIDT2 Controller

The PIDT2 controller is basically the same as the PIDT controller. However, it contains a second data input, which is scalable and will be added directly to the controller output signal. It therefore works as a general feed forward path. An additional filter in the input path can be activated for stabi- lizing the control loop.

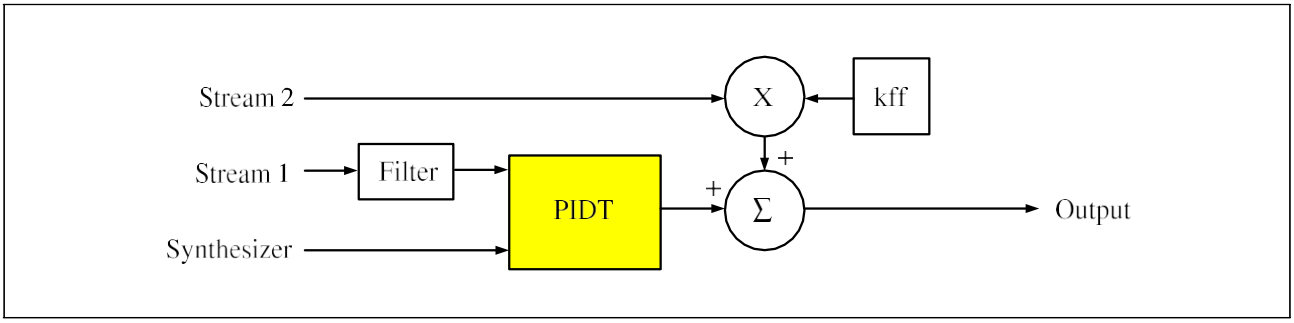


Figure 12: PIDT2 controller layout

Parameter Trimming

Table 18 lists the trimmable PIDT2 parameters.

Trimmable PIDT2 Parameter	Parameter Name	Possible Range
Controller gain	kp	0 ≤ kp
Integral part	ti	0 <ti

Table 18: Trimmable PIDT2 parameters and parameter names.

Trimmable PIDT2 Parameter	Parameter Name	Possible Range
Differential part	td	$0 \leq td$
First order differential part limiting(1)	t1	$T < t1 < td$
Feed forward gain	kff	$0 \leq kff$

Table 18: Trimmable PIDT2 parameters and parameter names.

1. T denotes the sampling interval.

CASCADE Controller

Der Cascade Controller ist in erster Linie für die Anwendung mit dem Servoverstärker im PCS8000- V3 gedacht. Es handelt sich dabei eigentlich um einen dreifach verschachtelten Regler, wobei der innerste Teil - der Stromregler - im Servoverstärker realisiert ist. Im mittleren Teil arbeitet ein PIDT- Algorithmus als Geschwindigkeitsregler. Diesem überlagert ist ein PIDV-Algorithmus als Position- sregler. Bei solchen überlagerten Regelstrukturen muss gewährleistet sein, dass die inneren Regler schneller arbeiten als die äusseren. Deshalb arbeitet der Stromregler im Servoverstärker mit vollem Regeltakt des PCS8000. Der Geschwindigkeitsregler läuft mit dem halben Regeltakt und der Posi- tionsregler schliesslich mit einem Viertel.
Abbildung 13 zeigt die Reglerstruktur, des im PCS8000 implementierten Teils. Daraus geht hervor,

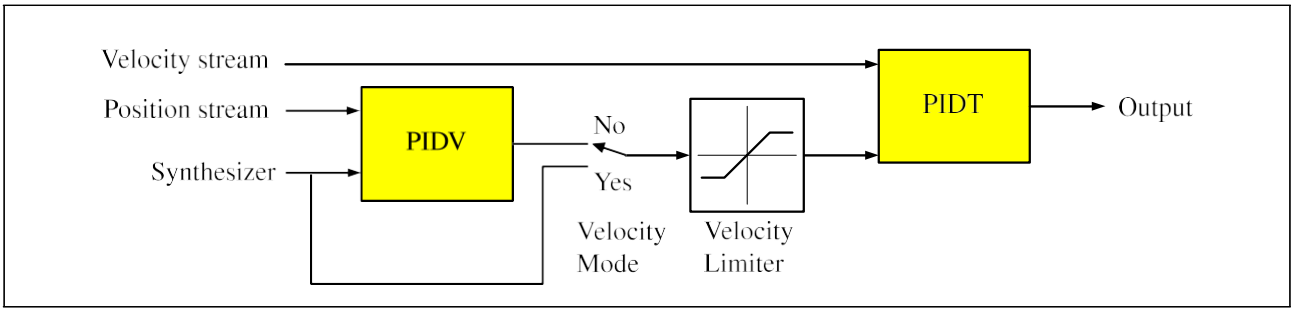


Figure 13: Cascade controller layout

dass:

- der Regler zwei Eingänge besitzt, den Positions- und den Geschwindigkeitsmesswert (beide werden vom Servoverstärker geliefert). Der primäre Eingang ist die Position, der sekundäre die Geschwindigkeit.
- der Positionsregler überbrückt werden kann. Dann arbeitet der Regler als reiner Geschwind- igkeitsregler. ACHTUNG: Dem primären Eingang muss dann die Geschwindigkeit zugew- iesen werden (der sekundäre Eingang wird nicht benötigt).
- der Geschwindigkeitsregler besitzt einen Sollwert-Limiter. Damit wird im Einrichtbetrieb (aktivierter Schlüsselschalter) der Sollwert der Geschwindigkeit auf einen, in der Systemkonfiguration einstellbaren Wert begrenzt.

Die Reglerstrukturen entsprechen den bereits vorgängig beschriebenen Strukturen PIDT und PIDV, wobei der hier verwendete PIDV-Algorithmus nur ein Subset der unter Kapitel 8.2 aufgelisteten Parameter unterstützt.

Folgende Tabelle zeigt die vom Cascade Controller verwendeten Parameter:

Cascade Controller Parameter	**Parameter Name**	**Wertebereich**	**Algorithmus**
Controller gain	kp	$0 \leq kp$	Velocity (PIDT)
Integral part	ti	$0 < ti$	Velocity (PIDT)
Differential part	td	$0 \leq td$	Velocity (PIDT)
First order differential part limiting[1]	t1	$T < t1 < td$	Velocity (PIDT)
Control mode	velocity_mode	no | yes	-
Quadratic gain	ke2	$0 \leq ke2$	Position (PIDV)
Feed forward gain	kff	$0 \leq kff$	Position (PIDV)
Controller gain	ke	$0 \leq ke$	Position (PIDV)
DC-Correction gain	kdcc	$0 \leq kdcc$	Position (PIDV)
Peakcontroller	pk	$pk = 0\ pk \geq 1$ [2]	Position (PIDV)
Gain for residual error integrator	krei	$0 \leq krei$	Position (PIDV)
Timeconstant for residual error integr.	tau	$0 \leq tau$	Position (PIDV)
Errorlimit to deplete rei	elim	$0 \leq elim$	Position (PIDV)
Manual balancer	sym_man	$0 \leq sym - man$	Position (PIDV)
Automatic balancer	sym_adp	0, 0.5...2[3]	Position (PIDV)
Kicker gain	gki	$0 \leq gki$	Position (PIDV)
Kicker timeconstant	tki	$0 \leq tki$	Position (PIDV)

Table 19: Cascade Controller Parameter

1. T denotes the sampling interval.
2. pk roughly denotes the number of periods which are required to reach the desired amplitude.
3. 0: disabled; 1: symmetrical behaviour

7.3. Drive Unit

The drive unit prepares the output signal before sending it to the drive. This includes mainly two tasks:

- Limit output level if required.
- Add dither signal by using the wave3 generator from the synthesizer.

8. Data Acquisition

The data acquisition contains several modules which either generate or consume data streams. They all run with the same sample rate. Figur 14 illustrates four of these modules:

- Synthesizer, generation of desired signals (constants, ramps, periodic functions, arbitrary curves and more).
- Measuring system for acquisition of sensor signals.
- Data exchange (useful in multi channel setup).
- Analog output, for driving servo valves or other actuators.

Figur 14 gives a detailed overview of the whole data acquisition and processing part. Most of it is realised in software. However, some parts, especially the data exchange uses hardware support. The data exchange, which is used in a multi channel setup is based on a proprietary high-speed serial link, which allows several controllers to exchange measuring data among each other within one sam- ple clock.

8.1. Sensor memory

Each external sensor is equiped with a EEPROM. These memories contain information about the sensor, its operating range, linearization, etc. At startup the central management module reads all these memories and builds up the sensor database with data gathered from the sensors (see chapter 4.2).

The sensor memory contains the following fields:

- Version number of sensor memory encoding (=> 2).
- Sensor type:

```
'None'          /* None. */
'Force'         /* Force. */
'Strain'        /* Strain, elongation. */
'Stroke'        /* Stroke, lift. */
'Acceleration'  /* Acceleratioin. */
'Temperature'   /* Temperature. */
'Humidity'      /* Humidity. */
'Light'         /* Light. */
'Voltage'       /* Voltage. */
'Current'       /* Current. */
'Power'         /* Power. */
```

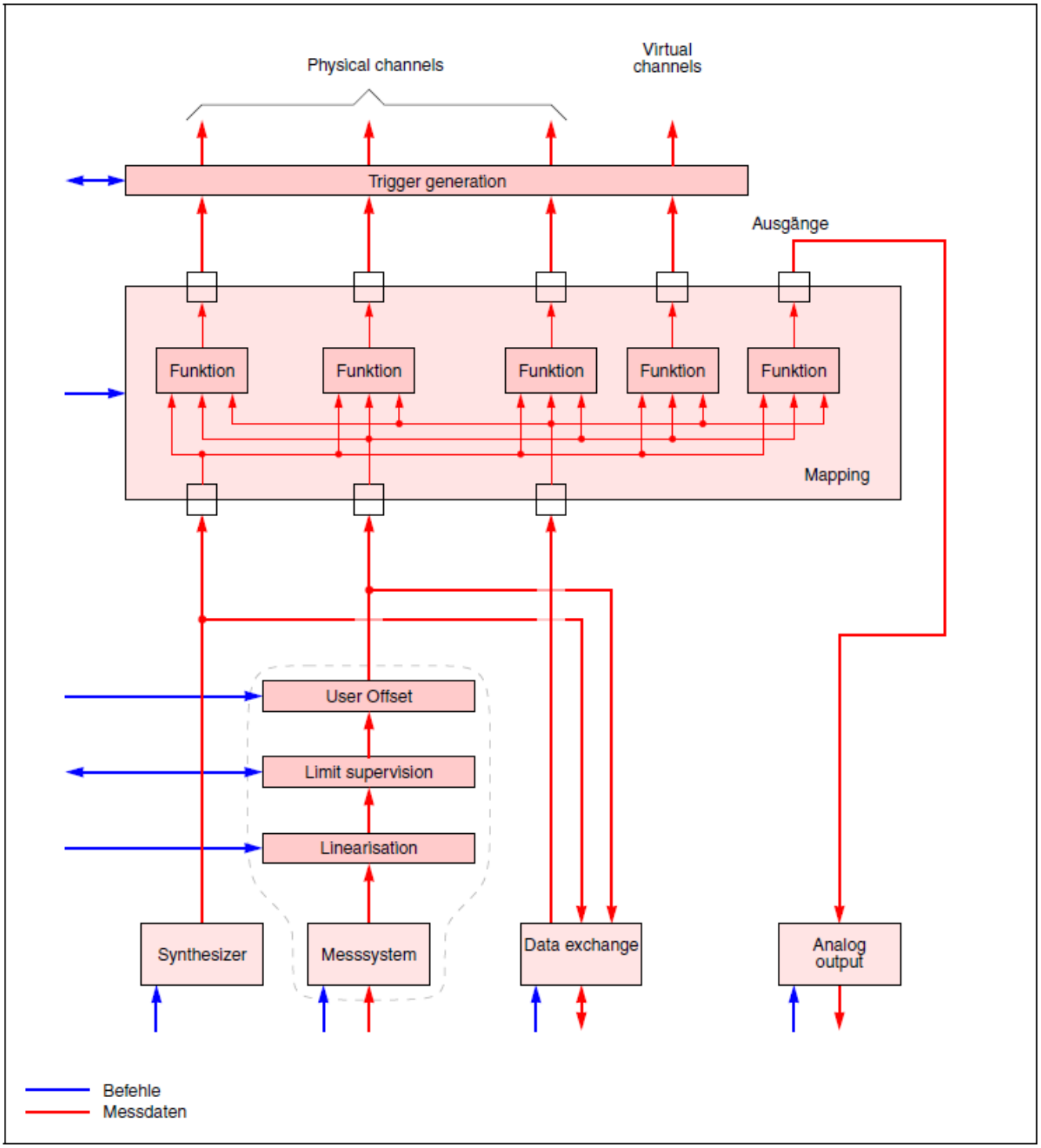


Figure 14: Detaildarstellung der Datenerfassung.

'Resistance'	/* Resistance. */
'Angle'	/* Angle. */
'Angle-Velocity'	/* Angle velocity. */
'Angle-Acceleration'	/* Angle acceleration. */
'Velocity'	/* Velocity. */

'Time'	/* Time. */
'Frequency'	/* Frequency. */

- Electrical amplifier type:

'NONE'	/* None. */
'DC'	/* DC type. */
'DMS'	/* DMS type. */
'LVDT'	/* LVDT type. */
'QUADENC_422'	/* Quadrature encoder RS-422. */
'QUADENC_TTL'	/* Quadrature encoder TTL. */
'INTERPOLATOR '	/* Analog with interpolator. */
'SSI '	/* Absolute through SSI. */
'Power'	/* Power. */

- Sensor name, ID, serial number and arbitrary description.
- Resolution for incremental position sensors.
- Sensor course sensitivity:

DMS type	DC type	LVDT type '8 mV/V'	'10 V'	'640 mV/V'
'4 mV/V	'5 V'	'320 mV/V'		
'2 mV/V'	'2.5 V'	'160 mV/V'		
'1 mV/V'	'1.25 V'	'80 mV/V'		
'0.5 mV/V'	---	---		
'0.25 mV/V'	---	---		

- Sensor fine sensitivity: a float value similar to above with a small deviation.
- Sensor excitation voltage for DC and DMS type sensors:
  '1 V'
  '2 V'
  '5 V'
  '10 V'

- Unit string of measured quantity (limited to 4 characters).
- Unit-ID number (0...65535, where 1. .....65534 can be assigned a literal string)
- Limits of operating range, tolerance and behaviour when in overload:

'Emergency stop'	/* Emergency stop.*/
'Standby'	/* Synthesizer stops at actual /* position and control loop will be. */ /* disabled */
'Setup mode'	/* Change to seq. state to setup. */
'Notify'	/* Simply send a message to host PC. */
'Nothing'	/* Do nothing at all. */

- SSI transfer rates:
  '1000 kHz'

'750 kHz'
'500 kHz'
'250 kHz'

- SSI number of bits: 8 to 32 Bits
- SSI data encoding:
  'Binary' /* Binary encoding (default). */ 'Gray' /* Gray encoding.*/

- Type of interpolation:
  'None' /* No interpolation. */
  'IPL200' /* IPL200 interpolation circuit. */

- Interpolation factors for IPL200 interpolation circuit:
  '20'
  '25'
  '40'
  '50'
  '80'
  '100'
  '160'
  '200'

- Type of linearization:
  'None' /* Linearization by a factor only*/
  'Polynomial' /* Linearization by polynomial. */

- Order of polynomial if linearization type is 'Polynomial'.
- Polarity (normal or inverted)
- Nominal load in physical quantities
- Positive and negative calibration pairs of ADC values and corresponding physical quantities.
- Lowpass filter setting, cutoff frequencies:
  "No filter"
  "Lowpass(1500Hz)"
  "Lowpass(1000Hz)"
  "Lowpass(500Hz)"
  "Lowpass(250Hz)"
  "Lowpass(125Hz)"
  "Lowpass(60Hz)"
  "Lowpass(20Hz)"

8.2. Linearisation

The linearisation unit provides a mapping from internal numbers to physical quantities. By means of a polynomial mapping it can also correct certain nonlinearities of the sensor.
It also provides a supervision of the given limits. If any limit is exceeded a message to the maincon- trol unit is issued.

8.3. User offset

The user can set and reset an individual offset value for each data stream. This is called the user off- set. It can be applied to any physical data stream, even on this which is in use by the control loop with the only restriction that the synthesizer has to be in constant mode.
Ab Firmware 2.8.0 unterstützen auch die virtuellen Kanäle einen User offset. Dieser wird gleich wie bei den physikalischen Kanälen pro Kanal hinzu addiert und zwar unmittelbar nachdem der neue vir- tuelle Wert berechnet wurde.
Die Arrays mit den User offset Werten sind auch direkt via Daten-Makros in virtuellen Kanälen und Berechnungen in Sequenzerprogrammen zugänglich (siehe folgende Tabelle).

8.4. Data exchange

In a multicontroller setup, each controller has the ability to export its streams and to import streams from other controllers. Data exchange relies on a realtime highspeed data link. Each controller can import 16 foreign streams at most. Data transport latency is constant at two samples.

8.5. Mapping and virtual channels

Trigger generation.
Programable.

8.6. Streams, Triggers and Variables

Table 20 lists all available data streams and related data variables.

Data	Stream Name	Data Variable Name	n
Sensor data(1)	phys1 ... phys15	DATA_PHY(n)	1 ... 15
Maximum drag indicator for phys1 ... phys15		DRAG_MAX_PHY(n)	1 ... 15
Minimum drag indicator for phys1 ... phys15		DRAG_MIN_PHY(n)	1 ... 15
Linearized sensor data(2)		DATA_PHY_RAW(n)	1 ... 15
Raw sensor data(3)		DATA_RAW(n)	1 ... 15

Table 20: Data variable names.

Data	Stream Name	Data Variable Name	n
Data values from history buffer		DATA_PHY_OLD(n,d) with d=1...4000	1 ... 15
Data slope values		DATA_PHY_SLOPE(n,d) with d=1...4000	1 ... 15
Synthesizer output	synth	SYNTH	
Maximum drag indicator for synth		DRAG_MAX_SYNTH	
Minimum drag indicator for synth		DRAG_MIN_SYNTH	
Synthesizer values from history buffer		SYNTH_OLD(d) with d=1...4000	
Synthesizer slope values		SYNTH_SLOPE(d) with d=1...4000	
Cycle counter for wave1		CYCLE1	
Cycle counter for wave2		CYCLE2	
Cycle counter for curve		CYCLE_CURVE	
Dither output stream	dither	DITHER	
Maximum drag indicator for dither		DRAG_MAX_DITHER	
Minimum drag indicator for dither		DRAG_MIN_DITHER	
Virtual data	virt1 ... virt32	DATA_VIR(n)	1 ... 32
Virtual values from history buffer		DATA_VIR_OLD(n,d) with d=1...4000	1 ... 32
Virtual slope values		DATA_VIR_SLOPE(n,d) with d=1...4000	1 ... 32
Maximum drag indicator for virt1 ... virt32		DRAG_MAX_VIR(n)	1 ... 32
Minimum drag indicator for virt1 ... virt32		DRAG_MIN_VIR(n)	1 ... 32
Controller output	control	CONTROL	
Controller values from his- tory buffer		CONTROL_OLD(d) with d=1...4000	

Table 20: Data variable names.

Data	Stream Name	Data Variable Name	n
Controller slope values		CONTROL_SLOPE(d) with d=1...4000	
Maximum drag indicator for control		DRAG_MAX_CONTROL	
Minimum drag indicator for control		DRAG_MIN_CONTROL	
External data	ext1 ... ext16	DATA_EXT(n)	1 ... 16
External data from history buffer		DATA_EXT_OLD(n,d) with d=1...4000	1 ... 16
External slope values		DATA_EXT_SLOPE(n,d) with d=1...4000	1 ... 16
Maximum drag indicator for ext1 ... ext 16		DRAG_MAX_EXT(n)	1 ... 16
Minimum drag indicator for ext1 ... ext 16		DRAG_MIN_EXT(n)	1 ... 16
Calculated values	calc1 ... calc16	DATA_CALC(n)	1 ... 16
Calculated values from history buffer		DATA_CALC_OLD(n,d) with d=1...4000	1 ... 16
Calculated slope values		DATA_CALC_SLOPE(n,d) with d=1...4000	1 ... 16
Maximum drag indicator for calc1 ... calc16		DRAG_MAX_CALC(n)	1 ... 16
Minimum drag indicator for calc1 ... calc16		DRAG_MIN_CALC(n)	1 ... 16
Acquisition time in ns		TIME	
Sequencer variables		VAR(n)	0...999[4]
State of wave generators (1 if any is running, 0 otherwise)		_AC_ON_	
State of curve generator (1 if a curve, aka a wave file, is being played)		_CURVE_ON_	
State of certain digital input (0 or 1)		DIGIN(n)	1..12

Table 20: Data variable names.

Data	Stream Name	Data Variable Name	n
Digital inputs as binary pattern		DIGIN_ALL	
Actual or last valid rate value		RAMP_ACTUAL_RATE	
User offset table for physi- cal channels		UOFS_PHY(n)	1..15
User offset table for vir- tual channels		UOFS_VIR(n)	1..32

Table 20: Data variable names.

1. Linearized sensor data with user offset and polarity correction (push-pull), available as stream and variable.
2. Linearized only sensor data (as output of the linearisation block according to Figure 14), available as vari- able only.
3. Raw data from the data acquisition part, available as variable only. The range of numbers depends on the input type. In case of an analog input with AD-converter it is $+-2^{23}$.
4. The index n defines the n-th variable from a sequencer program, starting with 0. The variables are enumer- ated in ascending order of their definition within the "vars"-tags of the loaded sequencer program. If no pro- gram is loaded, values of variables are undefined.

All values in the table above can be used in expressions for calculations within virtual channels and sequencer programs.

Table 21 lists all possible data acquisition events that can trigger actions in the sequencer. The table also includes some useful triggers from other software modules.

Event	Trigger Name	n
Maximum threshold of phys1 ... phys15	TRIG_MAX_PHY(n)	1 ... 15
Minimum threshold of phys1 ... phys15	TRIG_MIN_PHY(n)	1 ... 15
Ramp complete of synth	TRIG_SYNTH_R	
Wave 1 complete of synth	TRIG_SYNTH_W1	
Wave 2 complete of synth	TRIG_SYNTH_W2	
External stream complete of synth	TRIG_SYNTH_S	
Curve complete of synth	TRIG_SYNTH_C	
Maximum threshold of synth	TRIG_MAX_SYNTH	
Minimum threshold of synth	TRIG_MIN_SYNTH	

Table 21: Data acquisition triggers.

Event	Trigger Name	n
Dither complete of dither	TRIG_DITHER	
Maximum threshold of dither	TRIG_MAX_DITHER	
Minimum threshold of dither	TRIG_MIN_DITHER	
Maximum threshold of virt1 ... virt32	TRIG_MAX_VIR(n)	1 ... 32
Minimum threshold of virt1 ... virt32	TRIG_MIN_VIR(n)	1 ... 32
Maximum threshold of control	TRIG_MAX_CONTROL	
Minimum threshold of control	TRIG_MIN_CONTROL	
Maximum threshold of ext1 ... ext16	TRIG_MAX_EXT(n)	1 ... 16
Minimum threshold of ext1 ... ext16	TRIG_MIN_EXT(n)	1 ... 16
Maximum threshold of calc1 ... calc16	TRIG_MAX_CALC(n)	1 ... 16
Minimum threshold of calc1 ... calc16	TRIG_MIN_CALC(n)	1 ... 16
Timeout counter in ns[1]	TRIG_TIME	
Host trigger	TRIG_HOST(n)[2]	1 ... 16
Multi-controller synchronization	TRIG_SYNC	
Digital input triggers	TRIG_DIN(n)	1 ... 12
Digital out triggers	TRIG_DOUT(n)	1 ... 12
Eurotherm command complete (RS-485)	TRIG_EUROTHERM	

Table 21: Data acquisition triggers.

1. The time event triggers after a relative period of time, counting from the moment, when the trigger period was set.
2. TRIG_HOST can be used as a placeholder for TRIG_HOST(1)

8.7. Pull and push modes

Switching between pull and push mode is basically a matter of polarities. Usually a pulling force is represented by a positive value and a pushing force by a negative value. However, some people like to have positive values allways. They can switch to push mode when pushing forces arise. The polar- ity of all physical streams for both modes can be set independantly by the push-pull configuration.
The polarity of the controlling valve has to be aligned with the polarity of the controlling stream. For that purpose the valve polarity can be configured for both modes and for all possible controlling streams. These are not only the physical streams but also virtual and external streams.

Changing the push-pull configuration is permitted only in standby mode (with disabled controlloop) because otherwise dangerous situations could occur. This configuration is saved within a XML-file on the controller and will be resumed on each startup.
Switching between push and pull mode is possible allways except program mode. When a switch command is issued, the controller will set the correct polarities, reset all synthesizer generators and - if required - invert the internal state values of the control algorithm. All this is done synchronously within one sample to provide a smooth bumpless switchover.
In a multi channel setup each controller has its own push-pull configuration. However, the current push-pull mode is equal on all controllers. Therefore, the switching command is only allowed to be sent to the master. All slaves will switch concurrently with the master.
The current push-pull mode is memorized on the master and will be resumed on the next startup.
All commands concerning push-pull mode and configuration are addressed to the mapper unit. Refer to 'Mapping and Virtual Streams on page 118' for a description.

9. Commands

Commands reach the controller from outside or from inside. In both cases commands are handled and dispatched by the central management unit. Some commands are executed by the central man- agement itself, others are forwarded to their corresponding software unit.

9.1. Formats

Commands from outside of the controller are in XML format. Internal commands are pointers to data structures containing command details and parameters. The two command types are equivalent and each command can be translated into the other type.
Apart from explicit commands each software unit provides a function API that can be used by the central management and by other modules. Therefor, simple module interactions do not have to go through XML or data structure commands.

XML Commands
Commands in XML representation are structured as follows (EBNF):

< unit >
< command >
{ < parameter > value{[;value]} < |parameter > }
< |command >
< |unit >

For describing sequencer programs an extended structure will be used. Compared to the structure above it allows several commands within the unit tags:

< unit >
< command >
{ < parameter > value{[;value]} < |parameter > }

```
        <|command>

        {<command>
                {<parameter> value{[;value]} <|parameter>}
        <|command>}
<|unit>
```

The entire XML command is a pure ASCII string. Special characters like umlaute, accents, etc. are not allowed. Newline characters, spaces, etc. between tags are allowed but without meaning. Spaces between tags and values will be discarded as well. Only spaces within a parameter value will be con- sidered.
Tags are keywords enclosed in '<' and '>' characters. There are no spaces, newlines, etc. allowed within tags. Tags appear always in pairs, first a start tag, then an end tag. The end tag is marked with a '/' after the '<' character.
As shown in the example above XML commands are clearly structured in a kind of layers. The out- ermost layer addresses a controller module or unit. The second layer names the command and the third layer lists parameters, if any.
Normally only a single command is allowed within one XML command string. Multiple commands are only allowed when downloading a sequencer program (see chapter 10.16).
XML commands are always acknowledged. The reply is similar to the command but contains an additional <ackn> tag on the unit level. In an acknowledge telegram all command and parameter level tags are optional. Usually, if the command is not a 'get', it will be omitted. Here is an example:

```
<unit>
        [<command>
                {<parameter> value <|parameter>}
        <|command>

        {<command>
                {<parameter> value <|parameter>}
        <|command>}]

<ackn> OK <|ackn>
<|unit>
```

The <ackn> tag contains the value OKif everything is fine. Otherwise it contains the value ERROR followed by some optional string explaining the problem.

Internal Representation

The internal representation of commands is more machine-friendly. External XML messages will be converted by the communication modules. The equivalent data structure looks like the following.

Structur -- >	Functional Unit -- >	unit
	Number of Commands -- >	number
	Error message -- >	ackn
	Slave number -- >	slave
Command 0 -- >	Command Name -- >	command_0
	Number of Parameters -- >	number_0
	Name of Parameter 0 -- >	parameter_0_0
	Value of Parameter 0 -- >	value_0_0
	Name of Paremeter 1 -- >	parameter_0_1
	Value of Parameter 1 -- >	value_0_1
	...	...
Command 1 -- >	Command Name -- >	command_1
	Number of Parameters -- >	number_1
	Name of Parameter 0 -- >	parameter_1_0
	Value of Parameter 0 -- >	value_1_0
	Name of Paremeter 1 -- >	parameter_1_1
	Value of Parameter 1 -- >	value_1_1
	...	...
Command 2 -- >	Command Name -- >	command_2
	...	...

Table 22: Internal command data structure.

Values in the data structure, like unit name, command, parameter names, parameter values, etc. are just ASCII strings with the exception of the number of parameters. Parameter names, etc. can easily be identified over some hash functions. The type of a parameter value follows from the parameter name.

Master/Slave Communication

In a multi-controller setup the host PC only has a group 1 command interface to the master control- ler. This way the host PC only has to know the IP address of one controller, the master controller.

Commands addressed to a slave controller are sent to the master but include a <slave> tag directly after the <*unit*> tag.

```
        <unit>
<slave> nr </slave>
                <command>
                        { <parameter> value </parameter> }
                </command>
        </unit>
```

Whenever the master controller receives a command with a <slave> tag and *nr* > 0 (see table 3) from the controlling host PC (group 1 communication) it first checks if the corresponding slave con- troller is present. It is an error if that slave is not connected to the master by a group 3 communica- tion link. If the slave is connected the master forwards the entire command to that slave controller on the group 3 interface.
When the master controller receives a command with a <slave> tag and *nr* > 0 from a slave (group 3 communication) it forwards the entire telegram to the controlling host PC. Normally such telegrams are acknowledges to previous commands.

Telegrams without a <slave> tag or with *nr* = 0 are handled by the master controller whether they arrive by group 1 or group 3.

9.2. Central Management

System Configuration
Note that these commands are normally only used within the start script.
The name of the start script is start.xml. It's located within /etc/wbc/ and will be loaded and executed automatically upon system startup. This script contains various xml commands which are descripted within this chapter. Typically loading a configuration is done within the start script by the following command.

Run an configuration script containing XML statements:

```
<maincontrol>
        <read_file>
                <name> /path/filename.xml </name>
        </read_file>
</maincontrol>
```

The file filename.xmlcontains valid XML statements. The command read_filelets the cen- tral management open that file and execute all its commands. This might be useful for example to load an application specific default configuration. In this case the path is /etc/wbc/config.xml.Multi- ple such command files may be installed on the controller.
The system configuration (also called config script) might be changed by transfering the content of the previously mentioned file. This is done with the config-command:

```
<maincontrol>
```

<config>
<multi> ... </multi>
<master> ... </master>
<slavenr> ... </slavenr>
<total_slaves> ... </total_slaves>
<master_ip> ... </master_ip>

<valve_range> ... </valve_range> [<valve_pol> ... </valve_pol>](9)
<ecom_pw> ... </ecom_pw>
<ecom_check> ... </ecom_check>
<fadetime> ... </fadetime>
<start_delay> ... </start_delay>
<start_drive_limit> ... </start_drive_limit>
<vdrive_limit> ... </vdrive_limit>
<vdrive_limit_offset> ... </vdrive_limit_offset>
<speed_limit> ... </speed_limit>
<samplefreq> ... </samplefreq>

<spv_lswitch> ... </spv_lswitch> (obsolete on V2)
<spv_drvrdy> ... </spv_drvrdy> (obsolete on V2)
<spv_temp> ... </spv_temp>
<spv_templimit> ... </spv_templimit>
<spv_valvecur> ... </spv_valvecur>
<spv_door> ... </spv_door> (obsolete on V2)
<spv_drdelay> ... </spv_drdelay>
<clthrpos> ... </clthrpos>
<clthrneg> ... </clthrneg>
<clampch> ... </clampch>

<lswitch_action> ... </lswitch_action>

<control_type> ... </control_type>
<synth_stream> ... </synth_stream>
<control_src2> ... </control_src2>
<kp> ... </kp>
<ti> ... </ti>
<td> ... </td>
<t1> ... </t1>
<bal> ... </bal>
<f99> ... </f99>
<step_rate> ... </step_rate>
<ke> ... </ke>

9. This parameter is obsolete. It's function has been replaced by the push-pull configuration.

<ke2> ... </ke2>
<kff> ... </kff>
<krei> ... </krei>
<tau> ... </tau>
<elim> ... </elim>
<kdcc> ... </kdcc>
<sym_man> ... </sym_man>
<sym_adp> ... </sym_adp>
<kl> ... </kl>
<f0> ... </f0>
<f1> ... </f1>
<sdc> ... </sdc>
<gki> ... </gki>
<tki> ... </tki>
<pk> ... </pk>
<velocity_mode> ... </velocity_mode>
<filter> ... </filter>
<unipolar> ... </unipolar>

<dither_type> ... </dither_type>
<dither_freq> ... </dither_freq>
<dither_ampl> ... </dither_ampl>

<wheel_gain1> ... </wheel_gain1> (obsolete on V2)
<wheel_gain2> ... </wheel_gain2> (obsolete on V2)

</config>
</maincontrol>

The parameters are explained in table 23.

Parameter	Comment	Value
multi	Is it a multi-controller system?	yes | no
master	Is this a master in a multi-controller setup?	yes | no
slavenr	Slave number if this is not a master controller.	1 ... 100
total_slaves	Total number of slaves in a multi-controller setup.	1 ... 100
master_ip	IP of master in multi-controller setup.	Valid IP
valve_range	Valve current range (in mA).	20 ... 300
(valve_pol)	Drive output polarity (obsolete)	1 | -1

Table 23: System configuration parameters.

Parameter	Comment	Value
ecom_pw	Ethercom: Authentication code up to 20 chars.	string
ecom_check	Ethercom: Enable connection checking (group 1, 3)	yes | no
spv_lswitch	Supervisor: Are limit switches present?	yes | no
spv_drvrdy	Supervisor: Is a drive ready signal present?	yes | no
spv_temp	Supervisor: Enable thermal shutdown?	yes | no
spv_templimit	Supervisor: Limit for thermal shutdown.	20 ... 80
spv_valvecur	Supervisor: Monitor valve current.(1)	yes | no
spv_drdelay	Delay time for checking the ready signal (sec)	0 ... 10
spv_door	Supervisor: Monitor safety door.	yes | no
clthrpos	Clamp lock: Positive threshold.	float >= 0
clthrneg	Clamp lock: Negative threshold.	float <= 0
clampch	Clamp lock: Corresponding data stream.	see table 20
lswitch_action	Limit switches: Action to take when limit reached.	estop | setup | standby | notify
control_type	Setup mode: Default stroke controller.	pid | pidt | pidm | pidt_pk | pidt_pk2 | pidv | pidt2 | cascade
control_src2	Setup mode: Second controller input	see table 20
kp	Setup mode: Default PIDx controller, Kp.	float >= 0
ti	Setup mode: Default PIDx controller, Ti.	float > 0
td	Setup mode: Default PIDx controller, Td.	float >= 0
t1	Setup mode: Default PIDT controller, T1.	float >= 0
bal	Setup mode: Default PIDM controller, balance.	float >= 0
f99	Setup mode: Default PIDM controller, F_{99}	float > 0
step_rate	Setup mode: Default PIDT_pk contr., step_rate.	float > 0

Table 23: System configuration parameters.

Parameter	Comment	Value
ke	Setup mode: Default PIDV contr., controllergain.	float >= 0
ke2	Setup mode: Default PIDV contr., quadratic gain.	float >= 0
kff	Setup mode: Default PIDV contr., feedforward.	float >= 0
krei	Setup mode: Default PIDV contr., residual error integrator.	float >= 0
tau	Setup mode: Default PIDV contr., timeconstant for residual error integrator	float >= 0
elim	Setup mode: Default PIDV contr., Errorlimit for residual error integrator.	float >= 0
kdcc	Setup mode: Default PIDV contr., DC-correction.	float >= 0
sym_man	Setup mode: Default PIDV contr., Manual bal- ancer.	float >= 0
sym_adp	Setup mode: Default PIDV contr., Adaptive bal- ancer.	float == 0 | 0.5 <= float <= 2
kl	Setup mode: Default PIDV contr., Leckage.	float >= 0
f0	Setup mode: Default PIDV contr., Lower forcelimit for leckage compensatation.	float >= 0
f1	Setup mode: Default PIDV contr., Upper forcelimit for leckage compensation.	float >= f0
sdc	Setup mode: Default PIDV contr., Static drift com- pensation.	-1 <= float <= 1
gki	Setup mode: Default PIDV contr., Kickergain.	float >= 0
tki	Setup mode: Default PIDV contr., Timeconstant for kicker.	float >= 0
pk	Setup mode: Default PIDV contr., Peakcontroller.	float == 0 | float >= 1
dither_type	Setup mode: Dither waveform type.	sine | rect
dither_freq	Setup mode: Dither frequency.	float 10 ... 1000
dither_ampl	Setup mode: Dither amplitude.	float 0 ... 1
vdrive_limit	Setup mode: Limit drive output.	float 0 ... 1
vdrive_limit_o ffset	Setup mode: An optional offset to the parameter above (to handle a valve offset).	-0.06 ... +0.06

Table 23: System configuration parameters.

Parameter	Comment	Value
synth_stream	Setup mode: Input stream used by synthesizer.	see table 20
wheel_gain1	Hand wheel trim gain factor (position 10)	1 .. 100
wheel_gain2	Hand wheel trim gain factor (position 100)	1 .. 100

Table 23: System configuration parameters.

1. Disable valve current monitoring if the drive is not current controlled (for example not a valve).

When that command is used by the host PC the supplied parameter set will be loaded but the config script will not automatically be overwritten. The next time the controller starts the old values will be loaded again. The following command stores the actual parameter set to the config script.

```
<maincontrol>
    <config_save> </config_save>
</maincontrol>
```

Note: For safety reasons this command should be used with great caution. The default parameter set in the config file should guarantee a safe and stable operation of the testing machine. These parame- ters are always reloaded when entering the setup mode.
The following command can be used to read the actually loaded config parameters. The replay will contain all config parameters from table 23.

```
<maincontrol>
    <config_get> </config_get>
</maincontrol>
```

System States
Change system state:

```
<maincontrol>
    <set>
        <sys_state> state </sys_state>
    </set>
</maincontrol>
```

Valid system states are described in chapter 4.1. Note that according to figure 3 not every system state change is allowed. The central management checks the transition and responds with thefollow- ing error message if the required state change is not allowed. In this case the old system state remains.

```
<maincontrol>
    <set>
<sys_state> state </sys_state>
```

```
        </set>
        <ackn> ERROR: error string. </ackn>
    </maincontrol>
```

Request actual system state:

```
    <maincontrol>
        <get> </get>
    </maincontrol>
```

The controller answers as follows:

```
    <maincontrol>
        <get>
            <sys_state> state </sys_state>
        </get>
        <ackn> OK </ackn>
    </maincontrol>
```

Sensor Database

The sensor database includes information about up to 13 sensors. These data are read from the sensor memory and stored in the sensor database at startup (see chapter 4.2).

Update sensor database and setup system:

```
    <maincontrol>
        <setup> </setup>
    </maincontrol>
```

Note that the setup command will only be accepted in the startup and standby system state. Normally this command will only be used within the start script.

Request information about installed modules (measuring channels) and available sensors. The latter are those which have a plug containing a valid configuration:

```
    <maincontrol>
        <get_meas_inputs> </get_meas_inputs>
    </maincontrol>
```

The controller will respond with two vectors[10] of numbers corresponding to the available modules and sensors:

```
    <maincontrol>
        <get_meas_inputs>
                <channels> list of installed modules </channels>
                <active> list of available sensors </active>
    </get_meas_inputs>
```

10. A vector is list of numbers, separated by a ';'

```
    </maincontrol>
```

For active sensors, detailed information can be requested: (where number is 1 to 13)

```
    <maincontrol>
        <get_sensor>
            <sensor> number </sensor>
        </get_sensor>
    </maincontrol>
```

The controller answers as follows. Possible values for the parameters are given in chapter 9.1.

```
    <maincontrol>
        <get_sensor>
            <sensor> number </sensor>
            <phys_type>          ... </phys_type>
            <elec_type>          ... </elec_type>
            <name>               ... </name>
            <id>                 ... </id>
            <sernum>             ... </sernum>
            <description>        ... </description>
            <course_sens>        ... </course_sens>
            <fine_sens>          ... </fine_sens>
            <ex_volt>            ... </ex_volt>
            <polarity>           ... </polarity>
            <filter>             ... </filter>
            <unit>               ... </unit>
            <unit_id>            ... </unit_id>
            <limit_behav>        ... </limit_behav>
            <ovld_behav>         ... </ovld_behav>
            <adc_offset>         ... </adc_offset>
            <nom_load>           ... </nom_load>
            <limit_pos>          ... </limit_pos>
            <limit_neg>          ... </limit_neg>
            <pos_ovl_tol>        ... </pos_ovl_tol>
            <neg_ovl_tol>        ... </neg_ovl_tol>
            <resolution>         ... </resolution>
            <ssi_rate>           ... </ssi_rate>
            <ssi_sampling>       ... </ssi_sampling>
            <ssi_encoding>       ... </ssi_encoding>
            <ssi_nof_bits>       ... </ssi_nof_bits>
            <ipl_type>           ... </ipl_type>
            <ipl_factor>         ... </ipl_factor>
            <lin_type>           ... </lin_type>
            <poly_order>         ... </poly_order>
            <n_ppos>             ... </n_ppos>
            <n_pneg>             ... </n_pneg>
```

```
            <cal_raw_pos>      ... </cal_raw_pos>
            <cal_phys_pos>     ... </cal_phys_pos>
            <cal_raw_neg>      ... </cal_raw_neg>
            <cal_phys_neg>     ... </cal_phys_neg>
        </get_sensor>
        <ackn> OK </ackn>
    </maincontrol>
```

The following command creates backup files of the whole sensor database in XML style. It's mostly used internally:

```
    <maincontrol>
        <save_sensor_database> </save_sensor_database>
    </maincontrol>
```

Unit-ID lookup table

For the unit_id entry in the sensor database, a table can be loaded which assigns a literal string to each number. These strings show up in the webinterface sensor configuration part. This table is nor- maly stored on the system disk of the controller and loaded at boot time (using the maincontrol-
> read_file command).

```
<maincontrol>
    <set_unitid>
        <idX> name </idX>
            ...
        </set_unitid>
    </maincontrol>
```

X: A number between 1 and 65534

9.3. Ethernet Communication (Group 1)

Note that group 1 control commands are normally only used within the start script. Start group 1 communication service:

```
    <ethercom>
        <start> </start>
    </ethercom>
```

Start group 1 communication service and log all traffic into file:

```
    <ethercom>
        <start>
            <logfile> file </logfile>
        </start>
    </ethercom>
```

Stop group 1 communication service:

```
<ethercom>
    <stop> </stop>
</ethercom>
```

Connection check telegram:

```
<ethercom>
    <check> </check>
</ethercom>
```

9.4. Ethernet Communication (Group 2)

Initialize Service
This service requires no initialization because it is based on UDP.

Manage data streams
Register a data stream:

```
<udpxmit>
    <register>
        <stream> str </stream>
    </register>
</udpxmit>
```

str is a data stream name out of table 2. Choose a downsampling rate:

```
<udpxmit>
    <set>
        <down_sample> ds </down_sample>
    </set>
</udpxmit>
```

Normally data streams are available at the control rate which is actually something like 8000 sps. Hence registered streams provide approx. 8000 samples per second. This doesn't make much sense at slow processes. Therefor, with ds > 1 the data rate will be reduced. For example, if ds = 100, only every 100th sample will be transmitted.
Deregister data streams:

```
<udpxmit>
    <deregister>
        <stream> str </stream>
    </deregister>
</udpxmit>
```

str is again a stream name from table 2.

Start, stop and clear all transmissions:

```
<udpxmit>
    <start> </start>
</udpxmit>

<udpxmit>
    <stop> </stop>
</udpxmit>

<udpxmit>
    <clear> </clear>
</udpxmit>
```

With start all registered data transmissions are started synchronously. With stop they are stopped. The command clear deregisters all streams and stops transmissions.

9.5. Ethernet Communication (Group 3)

Start and Stop Service
Note that group 3 control commands are normally only used within the start script. Start group 3 communication service:

```
<slavecom>
    <start> </start>
</slavecom>
```

Start group 3 communication service and log all traffic into file:

```
<slavecom>
    <start>
        <logfile> file </logfile>
    </start>
</slavecom>
```

Stop group 3 communication service:

```
<slavecom>
    <stop> </stop>
</slavecom>
```

Connection check telegram:

```
<slavecom>
    <check> </check>
</slavecom>
```

9.6. Ethernet Communication (Group 4)

Note that group 4 control commands are normally only used within the start script. Start group 4 communication service:

```
<eventcom>
    <start> </start>
</eventcom>
```

Stop group 4 communication service:

```
<eventcom>
    <stop> </stop>
</eventcom>
```

Master/Slave Communication

In order to establish group 4 communication links the slaves in a multi-controller setup need to know the controlling host PC's IP address. The following command distributes the host's IP address. In a multi-controller setup the master sends this message to every slave after the host PC has connected to the master's group 1 service.

```
<eventcom>
    <host>
        <ip> addr </ip>
    </host>
</eventcom>
```

Parameter *addr* is the host's IP address. The default address 0.0.0.0 means that no host is present.

9.7. Imetrum Stream Protocol

Konfiguration

Die Konfiguration wird als XML-File auf dem Regler abgelegt und beim Booten geladen.

```
<imetrum_proto>
    <config>
        <port> 1234(11) </port>
        <ipaddr> a.b.c.d(12) </ipaddr>
        <com_debug> off | on </com_debug>
        <chan1_active> off | on </chan1_active>
        <chan1_stream> virtX(13) </chan1_stream>
        <chan1_interpolation> off | on </chan1_interp...>
```

11. Portnummer des Videomesssystems (typ. 1234)
12. IP-Adresse des Videomesssystems
13. Angabe des virtuellen Kanals, X = 1...32

```
                <chan1_scale> float </chan1_scale>
                <chan1_offset> float </chan1_offset>
                ...
                (Wiederholung der letzten 4 Zeilen für chan2, chan3, chan4 und chan5)
                ...
            </config>
        </imetrum_proto>
```

Abfrage der Konfiguration

```
        <imetrum_proto>
            <config_get> </config_get>
        </imetrum_proto>
```

Abspeichern der Konfiguration

```
        <imetrum_proto>
            <config_save> </config_save>
        </imetrum_proto>
```

9.8. RS-232 Communication

DOLI-Protocol

Konfiguration
Die Konfiguration wird als XML-File auf dem Regler abgelegt und beim Booten geladen.

```
        <serial_sensor>
            <config>
                <port> /dev/ttyS0(14) </port>
                <baud> 38400 | 57600 | 115200 </baud>
                <parity> none | odd | even </parity>
                <bits> 7 | 8 </bits>
                <protocol> doli </protocol>
                <sens1_active> on | off </sens1_active>
                <sens1_stream> virt1..32 </sens1_stream>
                <sens1_interpolation> on | off </sens1_interp...
                <sens1_scale> float </sens1_scale>
                <sens1_offset> float </sens1_offset>
                ...
                (Wiederholung der letzten 5 Zeilen für sens2, sens3, sens4)
                ...
```

14. The controllers RS232 UART-port (default).

```
        </config>
    </serial_sensor>
```

Abfrage der Konfiguration

```
    <serial_sensor>
        <config_get> </config_get>
    </serial_sensor>
```

Abspeichern der Konfiguration

```
    <serial_sensor>
        <config_save> </config_save>
    </serial_sensor>
```

9.9. HTTP Communication

The web interface can be enabled or disabled within the controller's config file or by an XML com- mand.

Enable interface

```
    <webif>
        <start> </start>
    </webif>
```

Disable interface

```
    <webif>
        <stop> </stop>
    </webif>
```

9.10. RS-485 Communication

Eurotherm series 2000

Get an operating value

```
    <eurotherm>
        <get>
            <devnr> 1...10 </devnr>
            <command> XX </command>
        </get>
        </eurotherm>
```

The device number is the number which has been assigned to a certain device within the configura- tion. It is linked to the address of the device. According to the supported EI-Bisynch protocol, the command consists of precisely two letters (case sensitive) and describes the parameter to be read.
The controller will respond as follows:

```
<eurotherm>
    <get>
        <devnr> 1...10 </devnr>
        <command> XX </command>
        <value> floating-point number </value>
    </get>
</eurotherm>
```

Set an operating value

```
<eurotherm>
    <set>
        <devnr> 1...10 </devnr>
        <command> XX </command>
        <value> floating-point number </value>
    </set>
</eurotherm>
```

Configuration

```
<eurotherm>
    <config_get> </config_get>
</eurotherm>
```

The controller will issue the configuration as follows:

```
<eurotherm>
    <config_get>
        <port> /dev/ttyS1(15) </port>
        <baud> 9600 | 19200 </baud>
        <parity> none | odd | even </parity>
        <bits> 7 | 8 </bits>
        <protocol> ei-bisynch | modbus(16) </protocol>
        <rate> 1..10(17) </rate>
        <dev1_address> 0..99(18) </dev1_address>
        <dev1_stream> virt1..virt32(19) </dev1_stream>
        <dev2_address>          ... </dev2_address>
        <dev2_stream>           ... </dev2_stream>
        <dev3_addres>           ... </dev3_address>
        <dev3_stream>           ... </dev3_stream>
        <dev4_address>          ... </dev4_address>
        <dev4_stream>           ... </dev4_stream>
        <dev5_address>          ... </dev5_address>
```

15. The controllers RS422/485 UART-port.
16. The current implementation only supports “ei-bisynch”, “modbus” is available as an upgrade.
17. The readout frequency in Hz.
18. The address of the Eurotherm device. 0 means device is disabled.
19. Data from Eurotherm devices is mapped to a certain virtual stream.

```
            <dev5_stream>          ... </dev5_stream>
            <dev6_address>         ... </dev6_address>
            <dev6_stream>          ... </dev6_stream>
            <dev7_address>         ... </dev7_address>
            <dev7_stream>          ... </dev7_stream>
            <dev8_address>         ... </dev8_address>
            <dev8_stream>          ... </dev8_stream>
            <dev9_address>         ... </dev9_address>
            <dev9_stream>          ... </dev9_stream>
            <dev10_address>        ... </dev10_address>
            <dev10_stream>         ... </dev10_stream>
        </config_get>
    </eurotherm>
```

By replacing “config_get” with “config”, the configuration can be written to the controller. After writing, the new configuration will be executed immediately, however, not stored. To store the con- figuration non-volatile on the controller, the following command must be used:

```
<eurotherm>
    <config_save> </config_save>
</eurotherm>
```

A stored configuration will be reloaded automatically at the next system start.

9.11. Digital I/O

The digital I/O subsystem provides a simple interface to the inputs and outputs. XML commands for external access as well as an internal function API exist.

Turn on and off

The following examples shows how to turn on a digital output:

```
<digital_io>
    <set>
        <output> n </output>
        <state> on </state>
    </set>
</digital_io>
```

For security reasons only the general purpose outputs from table 6 are allowed as *n*. Otherwise an error occurs. The following output states exist:

- Turn on: <state> on </state>
- Turn off: <state> off </state>

Request state

To check the state of an input or output use the following commands:

```
<digital_io>
    <get>
        <output> n </output>
    </get>
</digital_io>

<digital_io>
    <get>
        <input> k </input>
    </get>
</digital_io>
```

All possible number n and k from table 5 and 6 are allowed. The response will look like this:

```
<digital_io>
    <get>
        <output> n </output>
        <state> on </state>
    </get>
    <ackn> OK </ackn>
</digital_io>
```

In the response only the states on and off are possible.

Configuration

With the configuration of the Digital-IO unit, inputs and outputs(20) can be freely assigned to certain system functions. The configuration can be requested with the following command:

```
<digital_io>
    <config_get> </config_get>
<digital_io>
```

The controller will issue the configuration as follows:

```
<digital_io>
    <config_get>
    <drive_ready> ni </drive_ready>
    (21)[<upper_limit_switch> ni </upper_limit_switch>]
    (22)[<lower_limit_switch> ni </lower_limit_switch>]
    <P_Stop> ni </P_Stop>
    <N_Stop> ni </N_Stop>
    <remote_stop> ni </remote_stop>
    <upper_clamp_closed> ni </upper_clamp_closed>
    <lower_clamp_closed> ni </lower_clamp_closed>
```

20. All outputs except “Drive enable”, it’s fixed assigned to output 1.
21. Obsolete, replaced by <P_Stop> (will be read correctly for backward compatibility)
22. Obsolete, replaced by <N_Stop> (will be read correctly for backward compatibility)

```
<Pressure_system> ni </Pressure_system>
<Pressure_tapping_point> ni </...(23)
<Pressure_manifold> ni </...
<Leakoil_pump_ready> ni </...
<Pressure_traverse_clamp> ni </...
<test_running> no(24) </test_running>
<open_upper_clamp> no </open_upper_clamp>
<close_upper_clamp> no </close_upper_clamp>
<open_lower_clamp> no </open_lower_clamp>
<close_lower_clamp> no </close_lower_clamp>
<traverse_up> no </traverse_up>
<traverse_down> no </traverse_down>
<clamp_lock> no </clamp_lock>
<Fill_up_valve> nox </Fill_up_valve>
<Set_up_valve> nox </Set_up_valve>
<Pressure_relief_valve> nox </...
<Power_valve_1> nox </Power_valve_1>
<Power_valve_2> nox </Power_valve_2>
<Power_valve_3> nox <Power_valve_3>
<Leakoil_pump> nox </Leakoil_pump>
<Traverse_clamp_valve1> nox </...
<Traverse_clamp_valve2> nox </...
<Any_clamp_operation> nox </...
<Drive_on> no </Drive_on>(25)
<Setup_mode> no </Setup_mode>(26)
<Emergency_stop> no </Emergency_stop>(27)
<Variable_99x> no </...> (x: 0...9)(28)
    </config_get>
</digital_io>
```

ni: 1..12 when assigned to a digital input channel, 0 when not used.
no: 2..12 or HM1..HM15(29) when assigned to a digital output channel, 0 when not used. nox: HM1..HM15 when assigned to a digital output channel, 0 when not used.
By replacing “config_get” with “config”, the configuration can be written to the controller. After writing, the new configuration will be executed immediately, however, not stored. To store the con- figuration non-volatile on the controller, the following command must be used:

```
<digital_io>
```

23. Deprecated, but accepted for backward compatibility. Use “Pressure_manifold” instead.
24. Will get asserted while a sequencer program is running.
25. Will get asserted when drive is enabled (in parallel with output 1).
26. Will get asserted when setup mode is active (selected by key switch).
27. Will get asserted in any case of emergency stop (locally or remote).
28. Outputs controlled by variable values: =0:off, !=0:on.This works either when the value is set within a sequencer pro- gram (“update”-command) or when set by the host (see chapter 10.16, command “var_set”).
29. Output channels HM1..HM15 are available only when a hydraulic module is installed.

<config_save> </config_save>
</digital_io>

A stored configuration will be reloaded automatically at the next system start.
Remark: If group 4 communication is active, the host PC will be notified about state changes on dig- ital inputs and ouputs.

9.12. Start Script

There are no commands to the start script.

9.13. Remote Control (V1 controller)

In a multi-controller setup only the master controller has a remote control. In some cases the master's remote control should be used to trim parameters on a slave controller. In this case a command is used to tell the master to forward remote control events to the corresponding slave. The following command does exactly this.

<remote>
<forward>
<device> nr </device>
</forward>
</remote>

Device parameter value *nr* is a target slave number from table 3. If forwarding has been enabled by the above command the master controller will no longer respond to remote control events but send them, for example as trim commands, to the corresponding slave controller. To disable forwarding the above command is used with the master controllers device number (see table 3).
This command only applies to the PCS8000-V1!

9.14. Remote Control (V2 and V3 controller)

Konfiguration
Das Verhalten (Konfiguration) der Touch-Screen basierten Handsteuerung wird mit einem XML- Kommando fest gelegt. Obwohl nur der Master eine angeschlossene Handsteuerung hat, verfügen alle Regler in einem Mehrkanal-System über eine separate Konfiguration. Sobald auf der Hand- steuerung ein Slave ausgewählt wird, wird auch dessen Konfiguration geladen.
Die Konfiguration kann über die Weboberfläche editiert werden. Folgendes XML-Telegramm wird dabei verwendet:

<remote>
<config>
<language> ... </language>
<rc_nofstr> ... </rc_nofstr>
<rc_str1> ... </rc_str1>
<rc_str2> ... </rc_str2>

```
            <rc_str3> ... </rc_str3>
            <traverse> ... </traverse>
            <clamp> ... </clamp>
            <low_slope> ... </low_slope>
            <high_slope> ... </high_slope>
            <slope_dir> ... </slope_dir>
            <comspeed> ... </comspeed>
            <couple_devX> ... </couple_devX>
        </config>
    </remote>
```

Abfrage der Konfiguration

```
    <remote>
        <config_get> </config_get>
    </remote>
```

Als Antwort werden die Parameter aus folgender Tabelle geschickt.

Temporäres Ueberschreiben der Namen der Messkanäle

Die Namen der angezeigten Messkanäle auf der Fernsteuerung werden den Sensorsteckern entnom- men. Mit folgendem Telegramm können die angezeigten Namen temporär verändert werden. Aequivalent können auch die Namen und Einheiten von virtuellen Kanälen angegeben werden. Die Angaben, welche mit diesem Telegramm übermittelt werden, gehen bei einem Neustart verloren.

```
    <remote>
        <set>
            <physX_name> Name </physX_name>
            <virtX_name> Name </virtX_name>
            <virtX_unit> Einheit </virtX_unit>
            </set>
    </remote>
```

mit X = Nummer des angezeigten Messkanals.

Beschreibung der Parameter:

Parameter	**Beschreibung**	**Wert**
language	Spracheinstellung	"German" | "English" | "Cyrillic"
rc_nofstr	Anzahl angezeigter Messkanäle	0 ... 3
rc_str1	Messkanal 1	phys1 ... phys15 virt1 ... virt32
rc_str2	Messkanal 2	dito

Table 24: Parameter der RC-Konfiguration.

Parameter	Beschreibung	Wert
rc_str3	Messkanal 3	dito
traverse	Freigabe Traversenfunk- tion	“on” | “off”
clamp	Freigabe Klemmenfunk- tion	dito
low_slope	Geschwindigkeit für lang- sames Fahren	float
high_slope	Geschwindigkeit für schnelles Fahren	float
couple_devX(1)	Koppeln von Kanälen für simultanes Fahren im Mehrkanal-System	“no” | “yes”

Table 24: Parameter der RC-Konfiguration.

1. X ist die Device-Nummer, beginnend mit 0 für den Master. Dieser Parameter wird nur auf dem Master ausgewertet. Fehlt das Kommando für einen Regler, oder ist der Wert “no”, so gilt dieser Regler als nicht gekoppelt. Damit eine Kopplung stattfinden kann, muss für mindestens zwei vorhandene Regler der Wert auf “yes” stehen.

9.15. System Supervisor

Request supervisor status

```
<super>
    <get> </get>
</super>
```

Reply on status request:

```
<super>
    <get>
        <estop> ... </estop> [<eloop> ... </eloop>]
        <v1> ... </v1>
        <v2> ... </v2>
        <v3> ... </v3>
        <v4> ... </v4>
        <v5> ... </v5>
        <ref> ... </ref>
        <supply> ... </supply>
        <valve> ... </valve>
```

```
        <ctrvolt> ... </ctrvolt> (V2 only) [<link> ... </link>]
        <temp> ... </temp>
        <drvrdy> ... </drvrdy>
        <clamplock> ... </clamplock>
        <interpol> ... </interpol>
        <samplefreq> ... </samplefreq>
        <keyswitch> ... </keyswitch> (V2 only)
        <P_Stop> ... </P_Stop>
        <N_Stop> ... </N_Stop>
        <remstop> ... </remstop> (V2 only)
        <physX_state> ... </physX_state>
        <fw_version> ... </fw_version>
        <drv_version> ... </drv_version>
        <hal_version> ... </hal_version>
        <fpga_version> ... </fpga_version>
        <hw_version> ... </hw_version>
        <rc_version> ... </rc_version>
        <servofw_version> ... </servofw_version>
        <servohw_version> ... </servohw_version>
        <Pressure_system> ... </Pressure_system>
        <Pressure_tapping_point> ... </...
        <Leakoil_pump_ready> ... </Leakoil_pump_ready>
        <hydraulicmod> ... </hydraulicmod>
    </get>
</super>
```

Parameters are explained in table below:

Parameter	Comment	Value
estop	Emergencs stop switch	ok | stop
eloop[1]	Emergency stop loop	ok | stop
v1	Supply voltage 1, normally +5.00 V	float
v2	Supply voltage 2, normally +3.30 V	float
v3	Supply voltage 3, normally +1.20 V	float
v4	Supply voltage 4, normally +13.00 V	float
v5	Supply voltage 5, normally -13.00 V	float
ref	Reference voltage, normally +2.50 V	float

Table 25: System supervisor parameters.

Parameter	Comment	Value
supply	Common supply state	ok | error
valve	Valve current	ok | error
ctrvolt	Control voltage state (drive output)	ok | error
link[2]	High speed data link connection	ok | error
temp	Controller temperature (inside case)	float
drvrdy	Drive ready signal	on | off
clamplock	Clamp lock state	on | off
interpol	Stroke interpolator status signal	ok | error
keyswitch[3]	State of keyswitch (on remote control)	run | setup | error
P_Stop (vorher lswitch_up)[4]	Positive limit switch	on | off
N_Stop (vorher lswitch_lo)[5]	Negative limit switch	on | off
remstop	Remote stop signal (digital input)	on | off
physX_state[6]	State of physical data source (X: 1..13)	ok | limit | st_fail | vfault | cfault | over-load | error
samplefreq	Sampling frequency in Hz.	float
fw_version	Version der Firmware	Text
drv_version	Version des Treibers	Text
hal_version	Version der Hardware-Abstraction-Library	Text
fpga_version	Version des FPGA-Codes	Text
hw_version	Version des Mainboards	Text
rc_version	Version der Handrad-Firmware	Text
servofw_version	Version der Servoverstärker-Firmware	Text
servohw_version	Version der Servoverstärker-Hardware	Text

Table 25: System supervisor parameters.

Parameter	Comment	Value
Pressure_system(7)	Oeldruck am Eingang der Zapfstelle	ok | error
Pressure_tapping_p oint(7)	Oeldruck am Ausgang der Zapfstelle	ok | error
Leakoil_pump_ready (7)	Zustand Leckölpumpe	ok | error
hydraulicmod(7)	Zustand Hydraulikmodul	ok | error

Table 25: System supervisor parameters.

1. This parameter is transmitted only in a multi controller setup.
2. dito
3. Keyswitch states: run: normal state, setup: setup-mode only (starting sequencer programs denied), error: broken switch, indefinable state.
4. This parameter is transmitted only if there's a digital input assigned.
5. dito
6. For each installed module and the onboard channel there will be an entry which describes its state. The following states are possible: ok; limit(programmed sensor limit is reached); st_fail(selftest has failed); vfault(bridge voltage fault); cfault(bridge current fault); overload(AD converter over- load); error(general status fault). Remark: The states with a configurable behaviour are reported only if the behaviour is set to something else than none.
7. Diese Parameter werden nur übermittelt, wenn ein Hydraulikmodul eingebaut ist.

Set probe limit values and behaviour

```
<super>
    <probe_limit_set>
        <streamX_max> ... </streamX_max>
        <streamX_min> ... </streamX_min>
        <streamX_action> ... </streamX_action>
        <virtY_max> ... </virtY_max>
        <virtY_min> ... >/virtY_min>
        <virtY_action> ... </virtY_action>
        <clip_limit> ... </clip_limit>
    </probe_limit_set>
</super>
```

X: The stream number (1..15).
Y: Virtual channel number (1..32).
The parameters within this command can be sent in any order. The command doesn't have to be complete. Omitted parameters keep their current value. The controller stores all parameters in non- volatile memory and restores them after a power cycle.
Request probe limit values, behaviour and current state:

```
<super>
    <probe_limit_get>
```

```
            <streamX_max> ... </streamX_max>
            <streamX_min> ... </streamX_min>
            <streamX_action> ... </streamX_action>
            <streamX_state> ... </streamX_state>
            <virtY_max> ... </virtY_max>
            <virtY>_min> ... </virtY_min>
            <virtY_action> ... </virtY_action>
            <virtY_state> ... </virtY_state>
            <clip_limit> ... </clip_limit>
        </probe_limit_get>
    </super>
```

Parameter	Comment	Value
streamX_max	The maximum physical probe limit value	float
streamX_min	The minimum physical probe limit value	float
streamX_action	Action to take when limit reached.	estop | standby | setup |notify | none
streamX_state(1)	The state of a certain stream.	ok | limit
virtY_max	The maximum virtual probe limit value	float
virtY_min	The minimum virtual probe limit value	float
virtY_action	Action to take when limit reached.	estop | standby | setup |notify | none
virtY_state	The state of a certain stream.	ok | limit
clip_limit	Defines the value to which the control signal will be clipped when either a probe limit, a sensor limit or a limit switch event occurs. This value should be larger than the valve offset to prevent from oszillation.(2)	float (0..1) Default: 0.05

Table 26: System supervisor parameters for probe limit.

1 A stream with an action set to nonewill allways respond with ok.
2 This applies only when the behaviour is set to “setup mode”.

Bemerkung:
“probe_limit_get” liefert nur Angaben für aktuell konfigurierte Kanäle.

9.16. Sequencer

Change Sequencer State

```
<sequencer>
    <set>
        <seq_state> setup | program </seq_state>
    </set>
</sequencer>
```

A change to 'program' state immediately starts the preconfigured program. A change to setup state from program state terminates the program. Whenever a command to enter 'setup' state is received, the synthesizer will be reset and the controller is loaded with the parameters given in the system con- figuration.

Set Parameters for Setup Mode
Synthesizer:

```
<sequencer>
    <setup_synth_set>
        <unit> ramp|wave1|wave2|wave3|stream|curve </unit>
        <type> wavetype </type>
        <pulse> no | yes </pulse>
        <stream> str </stream>
        <stream2> str </stream2>
        <rate> ra </rate>
        <end_rate> er </end_rate>
        <acc> ac </acc>
        <freq> fr </freq>
        <start_freq> sfr </start_freq>
        <freqswp_type> none|lin|log|mod_wave1|mod_wave2|
        mod_wave3|mod_stream|mod_curve </freqswp_type>
        <freqswp_rate> frate </freqswp_rate>
        <ampl> am </ampl>
        <start_ampl> sam </start_ampl>
        <amplswp_type> none|lin|log|mod_wave1|mod_wave2|
        mod_wave3|mod_stream|mod_curve </amplswp_type>
        <amplswp_rate> arate </amplswp_rate>
        <duty> du </duty>
        <rise> ri </rise>
        <fall> fa </fall>
        <phase> ph </phase>
        <gain> gn </gain>
        <plimit> pl </plimit>
        <nlimit> nl <nlimit>
        <desired_stream> dstr </desired_stream>
```

```
                <file> file </file>
                <end_ampl> ea </end_amplitude>
                <end_time> et </end_time>
                <end_cycles> ec </end_cycles>
        </setup_synth_set>
</sequencer>
```

The Parameters *ra, er, ac, fr, sfr, am, sam, du, ri, fa, ph, ea, et, gn, pl* and *nl* are real values. Parameter ec is an integer type value. The synthesizer *wavetype* is a waveform name out of table
The data stream str and the *desired_stream* are stream names out of table 2. The filename
file is the name of a wave-file and applies only to the curve generator.

The parameter *stream* describes the controlled data channel and should be provided only when the controlled channel is to be changed. All synthesizer generators will be reset upon receiving this parameter.

The parameter *stream2* defines the second data channel and is required only for controllers with two inputs. It must be provided only when the channel is to be changed. This parameter doesn't influence the synthesizer functions, hence it can be sent with every senthesizer command,. However, it's recommended to send it together with the *stream* parameter.

The parameter *desired_stream* describes the input data channel for the external desired value generator.
The parameters *ampl_start* and *freq_start* define the starting values used for the sweep function. The corresponding end values are given be *ampl* and *freq*.
The parameter *amplswp_type* defines wether sweep or modulation for the Amplitude is to be used. The same applies to the frequency with *freqswp_type*.
The parameters *amplswp_rate* and *freqswp_rate* define the rate of change when using the sweep- function.

Important note: Giving an *end_time* parameter to a periodic function can be critical, since the out- put of the wave generator immediately jumps to 0 when the generator stops. So this is probably not what you want and generally it doesn't makes sense to use this parameter with periodic functions.

Control system:

```
<sequencer>
        <setup_control_set>
                <type>pid | pidt | pidm | pidt_pk | pidt_pk2 |
                        pidv | pidt2 | cascade </type>
                <kp> kp </kp>
                <ti> ti </ti>
                <td> td </td>
                <t1> t1 </t1>
                <bal> bal </bal>
                <t99> t99 </t99>
                <step_rate> rate </step_rate>
                <ke> ke </ke>
```

```
            <ke2> ke2 </ke2>
            <kff> kff </kff>
            <krei> krei </krei>
            <tau> tau </tau>
            <elim> elim </elim>
            <kdcc> kdcc </kdcc>
            <sym_man> sys_man </sym_man>
            <sym_adp> sys_adp </sym_adp>
            <kl> kl </kl>
            <f0> f0 </f0>
            <f1> f1 </f1>
            <sdc> sdc </sdc>
            <gki> gki </gki>
            <tki> tki </tki>
            <pk> pk </pk>
            <filter> filter_string </filter>
            <unipolar> no | yes </unipolar>
            <velocity_mode> no | yes </velocity_mode>
        </setup_control_set>
    </sequencer>
```

The control parameters *kp* > = 0, *ti* > 0, *td* > = 0 and T < *t1* < *td* are real values, T is the sampling interval. The additional PIDM parameters are 0 < *bal* < 1 and 0 < *t99*. The 'step_rate' is used by the PIDT_pk algorithm only and gives the adaption speed in % per second. The PIDV parameters are explained in Table 23 or more detailed in the report "UN110826_PIDV-Reglerdoku.pdf" (Version 0.97).
The parameter *filter_string* describes an optional filter within the PIDT2 controller. If this parameter is omitted then no filter used. The following filters are available:

- "No filter"
- "Lowpass(1500Hz)"
- "Lowpass(1000Hz)"
- "Lowpass(500Hz)"
- "Lowpass(250Hz)"
- "Lowpass(125Hz)"
- "Lowpass(60Hz)"
- "Lowpass(20Hz)"

Drive unit:

```
    <sequencer>
        <setup_drive_set>
            <limit> Ii </limit>
        </setup_drive_set>
    </sequencer>
```

Parameter *li* is a real value between zero and one. It limits the drive output level. For example if *li* = 0.4 the output level is limitted to +/- 40 %.
Start control loop with new parameters in 'setup' mode. Activate previously set parameters on syn- thesizer and controller:

```
<sequencer>
    <setup_start> </setup_start>
</sequencer>
```

Trim Synthesizer and Control Parameters
Parameter trimming is possible in 'setup' and 'program' mode. There is a setup command for trim- ming and the actual trim command.
Setup Synthesizer Trimming:

```
<sequencer>
    <synth_trim_set>
        <unit> ramp | wave1 | wave2 | wave3 </unit>
        <param> pa </param>
        <step> st </step>
        <min> mi </min>
        <max> ma </max>
    </synth_trim_set>
</sequencer>
```

pa is a name of a valid trimmable parameter. They are listed in table 11 to Table 17. Parameters *st*, *mi* and *ma* are real values. step means the stepsize per increment. That is the value that will be added to or subtracted from the parameter value for each increment. min and max are lower and upper limits of the trimmable range. Hence the parameter value will remain within these limits.
Setup controller trimming:

```
<sequencer>
    <control_trim_set>
        [<type> pid | pidt | pidm | pidt_pk | pidt_pk2 |
                pidv </type>]
        <param> pa </param>
        <step> st </step>
        <min> mi </min>
        <max> ma </max>
    </control_trim_set>
</sequencer>
```

The typeparameter is useless and might be omitted. *pa* is the name of a valid trimmable parame- ter. Table 13 to 17 list these parameters and parameter names.

Send increments to the sequencer:

```
< sequencer >
    < trim >
        < inc > N < /inc >
    < /trim >
< /sequencer >
```

Parameter value *N* is the number of increments. It can be negative or positive. Depending on the last trim set command the increments are sent to the synthesizer or the controller. If the given minimum or maximum values are reached, trimming will be stopped automatically.

Synthesizer Pause

At any time, in setup mode or in program mode, the user can interrupt the current procedure by paus- ing the synthesizer. Note: The user can not pause a sequencer program because this could lead to ambiguous situations with events, synchronization, etc. but he can pause the synthesizer which is safe.

The following command puts the synthesizer into pause mode and back.

```
< sequencer >
    < synth_pause > < /synth_pause >
< /sequencer >

< sequencer >
    < synth_cont > < /synth_cont >
< /sequencer >
```

Cycle Counters

Reset cycle counters to zero.

```
< sequencer >
    < clr_cycle >
        < unit > wave1 | wave2 | curve < /unit >
    < /clr_cycle >
< /sequencer >
```

Setzen und Lesen von Sequencer-Variablen

Variablen, welche innerhalb eines Sequencer-Programmes verwendet werden, können zu jederbelie- bigen Zeit gelesen oder geschrieben werden. Der Zugriff erfolgt allerdings nicht über deren Namen (siehe unten), sondern über einen Index, welcher der Reihenfolge der Aufzählung entspricht, begin- nend mit 0.

Lesen einer Variablen:

```
< sequencer >
```

```
        <var_get>
            <index> 0...999 </index>
        </var_get>
    </sequencer>
```

Antwort:

```
    <sequencer>
        <var_get>
            <index> 0...999 </index>
            <value> float-Zahl </value>
        </var_get>
    </sequencer>
```

Schreiben einer Variablen erfolgt äquivalent:

```
    <sequencer>
        <var_set>
            <index> 0...999 </index>
            <value> float-Zahl </value>
        </var_set>
    </sequencer>
```

Load a Sequencer Program

A statement to load a sequencer program has the following shape.

```
    <sequencer_prog>
    ...
    </sequencer_prog>
```

A sequencer program can only be downloaded while no other program is running. Loading a new program immediately destroys an existing program even if program checking detects program errors. A change into 'program' mode and hence starting a program is only possible after successfully downloading a program. Otherwise the state change will result in an error message.

The content of the sequencer program is a series of XML-commands. The following commands exist.

Define variables, expressions and conditions:

```
        <definitions>
            <vars> v_list </vars>

            <expr1> ex_1 </expr1>
            <expr2> ex_2 </expr2>
            <expr3> ex_3 </expr3>
            ...
            <cond1> co_1 </cond1>
```

```
        <cond2> co_2 </cond2>
        <cond3> co_3 </cond3>
        ...
    </definitions>
```

The definition statement has to be the first statement in a program.
v_list is a semicolon separated list of up to 1000 variable names. Variable names can include alpha-numeric ASCII characters, underscores, but no white spaces, no special char- acters, no umlaute, etc.

The tags <expr1> to <expr1000> define up to 1000 different expressions. Expressions are valid C-style assignments that can include user defined variables from *v_list* and data variables from table 20.

The tags <cond1> to <cond1000> define up to 1000 C-style expressions that result in a boolean value. Apart from user variables according to *v_list* and data variables from table 20 relational operators can be used. The following relational operator are supported:

eq	-	equal
lt	-	less than
le	-	less than or equal
gt	-	greater than
ge	-	greater or equal

Note that all variables and expressions are real valued. Hence be careful when using eq. The preferred relational operators are lt or le and gt or ge.
Here is an example:

```
<definitions>
    <vars> cnt; test </vars>
    <expr1> cnt = cnt+1 </expr1>
    <expr2> VAR(1)=10 </expr2> (is equal to test=10)
    <expr14> test = pow(DATA_PHY(2), 2) - 1 </expr14>
    <cond4> cnt ge 10 </cond4>
    <cond9> sin(test) lt 0.1 </cond9>
</definitions>
```

Define a label:

```
<label>
    <name> string </name>
</label>
```

A label can be used to identify a jump target location. Alpha-numeric ASCII characters and underscores are allowed, but no white spaces, no special characters, no umlaute, etc.
Update an expression previously defined in a <definitions> statement.

```
<update>
    <expressions> e_list </expressions>
</update>
```

e_list is a list of expression names like expr1, expr2, etc. Only the selected expres- sions will be updated. The updated expressions will be sent to the controlling host by a group 4 event telegram (see table 4).
Wait for a trigger event or state:

```
<wait>
    <triggers> t_list </triggers>
</wait>
```

When a <wait> statement within a program is encountered, program execution will be sus- pended until an expected trigger event or state from t_list will be met. If the expected trigger state is already present (triggered earlier) or as soon as one of the expected trigger events appear, program execution resumes. *t_list* is a semicolon separated list of trigger names according to table 21.
Jump to a label:
Jumps can be conditional or unconditional. A jump is conditional if a <triggers> and/or <conditions> tag is supplied.

```
<goto>
    <label> name </label>
    <triggers> t_list </triggers>
    <conditions> c_list </conditions>
</goto>
```

The label name *name* must be defined within the program. *t_list* is a list of trigger names according to table 21. *c_list* is a list of conditions like cond1, cond2, etc. previously defined in a <definitions> statement.
Clear a trigger state, drag indicator or cycle counter:

```
<clear>
    <triggers> t_list </triggers>
    <pdrag> d_list </pdrag>
    <ndrag> d_list </ndrag>
    <cycle> wave1 | wave2 </cycle>
</clear>
```

Trigger events remain pending until they are manually cleared. This command is used to clear triggers as well as the positive and negative drag indicators. *t_list* is a list of semico- lon separated trigger names from table 21. *d_list* is a semicolon separated list of stream names from table 20. <pdrag> lists the streams for which to clear the positive dragindica- tors and <ndrag> the ones to clear the negative drag indicator.
Send a string and/or value to the controlling host PC:

```
<host_send>
    <string> str </string>
</host_send>
```

This command sends string *str* as a group 4 event telegram. *str*may consist of any char- acters up to 512 Byte. Note that variables within the sequencer program are automatically sent to the host PC by group 4 communication whenever they are updated.
Receive a value from the controlling host PC after a host event:

```
<host_rec>
    <var> v </var> [<nr> 1...16 </nr>]
</host_rec>
```

With a host event the controlling PC can also send a value. With the previous command the sequencer program can assign that value to variable *v*.
Es können bis zu 16 unterschiedliche Werte gesendet werden. Ueber den optionalen Parame- ter <nr>können diese unterschieden werden. Wird <nr>weg gelassen, so wird 1 angenom- men. (siehe auch <host_trig>).
Setup synchronization of multiple controllers:

```
<sync>
    <device> list </device>
</sync>
```

The <sync>command does two things: (1) the controllers own sync flag is set immediately.
(2) the sync event (see table 21) is programmed to wait for all sync flags of the controllers in *list*. *list*is a semicolon separated list of slave numbers (see table 3).
Clear synchronization flag:

```
<unsync> </unsync>
```

Normally, after a successful synchronization, each controller involved would like to reset its sync flag. The <unsync>command does that.
A typical synchronization sequence looks like follows.

```
...
<wait> ... </wait>
<sync>
    <device> 0; 1; 2 </device>
</sync>
<wait>
    <triggers> TRIG_SYNC </triggers>
</wait>
<unsync> </unsync>
```

Preconfigure synthesizer:
For details see setup mode above.

```
<synth_set>
    <unit> ... </unit>
    <type> ... </type>
```

<pulse> ... </pulse>
<stream> ... </stream>
<stream2> ... </stream2>
<rate> ... </rate>
<end_rate> ... </end_rate>
<acc> ... </acc>
<freq> ... </freq>
<start_freq> ... </start_freq>
<freqswp_type> ... </freqswp_type>
<freqswp_rate> ... </freqswp_rate>
<ampl> ... </ampl>
<start_ampl> ... </start_ampl>
<amplswp_type> ... </amplswp_type>
<amplswp_rate> ... </amplswp_rate>
<duty> ... </duty>
<rise> ... </rise>
<fall> ... </fall>
<phase> ... </phase>
<gain> ...</gain>
<plimit> ... </plimit>
<nlimit> ... <nlimit>
<desired_stream> ... </desired_stream>
<file> ... </file>
<end_ampl> ... </end_ampl>
<end_time> ... </end_time>
<end_cycles> ... </end_cycles>
</synth_set>

Important note: Don't provide the 'stream'-tag when setting up a wave generator if the cur- rent stream source won't be changed. This would lead to a previous reset of the synthesizer.

Preconfigure controller:
For details see setup mode above.

<control_set>
<type> ... </type>
<kp> ... </kp>
<ti> ... </ti>
<td> ... </td>
<t1> ... </t1>
<bal> ... </bal>
<f99> ... </f99>
<step_rate> ... </step_rate>
<ke> ... </ke>
<ke2> ... </ke2>

<kff> ... </kff>
<krei> ... </krei>
<tau> ... </tau>
<elim> ... </elim>
<kdcc> ... </kdcc>
<sym_man> ... </sym_man>
<sym_adp> ... </sym_adp>
<kl> ... </kl>
<f0> ... </f0>
<f1> ... </f1>
<sdc> ... </sdc>
<gki> ... </gki>
<tki> ... </tki>
<pk> ... </pk>
<filter> ... </filter>
<unipolar> ... </unipolar>
<velocity_mode> ... </velocity_mode>
</control_set>

Start control loop:
The following command starts the control loop with the preset control and synthesizer parameters.

<start> </start>

Preconfigure drive unit:
For details see setup mode above.

<drive_set>
<limit> ... </limit>
<drive_set>

Configure data acquisition: For details see chapter 10.21.

<uoffset_set>
<stream> ... </stream>
<value> ... </value>
</uoffset_set>

Configure triggers:

<trigger_set>
<name> ... </name>
<value> ... </value>
</trigger_set>

Parameter <name> is a trigger name from table 21. <value> is the trigger threshold in

physical units of the corresponding data stream. These thresholds can be changed at runtime of the sequencer program. Commands from the mapper are used for this purpose. For details see chapter 10.21.
Switch digital output or relays: For details see chapter 10.11.

```
<digital_io>
    <output> ... </output>
    <state> ... </state>
</digital_io>
```

Note: The relays changes its state with a certain delay. Normally it switches during the next <wait> interval.
Handle data streams:
For details see chapter 10.4.

```
<udp_register>
<stream> ... </stream>
</upd_register>

<udp_start> </udp_start>

<udp_stop> </udp_stop>

<udp_clear> </udp_clear>
```

Get a process value from a Eurotherm device: For details see chapter 5.8.

```
<eurotherm_get>
    <devnr> 1...10 </devnr>
    <command> XX </command>
    <var> variable name </var> [<trig> yes </trig>]
<eurotherm_get>
```

Set a process value of a Eurotherm device: For details see chapter 5.8.

```
<eurotherm_set>
    <devnr> 1...10 </devnr>
    <command> XX </command>
    <value> variable name or float number </value> [<trig> yes </trig>]
    <eurotherm_get>
```

Since eurotherm commands are executed in a asynchonous manner, a trigger can be issued when the command is complete. This can be used to synchronize the program flow. Refer to

Table 21.

Send value to the Sequencer Program

```
<sequencer>
    <host_trig>
        <value> ... </value>
        [<nr> 1...16 </nr>]
    </host_trig>
</sequencer>
```

The controlling host PC can send an event trigger to the sequencer program. With the event the con- trolling PC can also send a value. The sequencer program can synchronize on that trigger with the <wait> statement. By using the <host_rec> command the sequencer program can assign that value to a variable.
Ueber den optionalen Parameter <nr> können bis zu 16 verschiedene Trigger und Werte gesendet werden. Wird <nr> weg gelassen, so wird 1 angenommen.

9.17. Synthesizer

Because the synthesizer is totally controlled by the sequencer it does not have its own XML inter- face.

9.18. Controllers

Because the controllers are totally controlled by the sequencer they do not have their own XML interfaces.

9.19. Drive Unit

The drive unit does not have its own XML interface. See sequencer commands in chapter 10.16.

9.20. Analog Output Channels

The analog output module provides four independant output voltages. The four channels A to D can be mapped to certain streams (see Table 2). Each channel provides an ouput voltage between -11.5 and +11.5V. Stream data is mapped to the output voltage by a linear function given through a positive and a negative corner point. The settings are memorized non-volatile on the module. The commands below describe how to request and change these settings.

Request current settings

```
<analogout>
    <get>
        [<module> 1 .. 4 </module>]
        <channel> A | B | C | D </channel>
</get>
```

```
</analogout>
```

The 'module' parameter is for future use and might be omitted. The controller responds with the following answer:

```
<analogout>
    <get>
        <module>1</module>
        <channel>ch</channel>
        <type> off | man | auto </type>
        <stream> st </stream>
        <pphys> float </pphys>
        <nphys> float </nphys>
        <pvolt> float </pvolt>
        <nvolt> float </nvolt>
    </get>
    <ackn>OK</ackn>
</analogout>
```

If the 'type' parameter is 'off', the output voltage is clamped to zero. If it's set to 'man' (manual), the mapping function is calculated using the given points (nphys/nvolt, pphys/pvolt). In auto mode the mapping is done automatically using the sensor limits, which are mappped to +-10V. This is obvi- ously possible only, if the stream refers to a configured (physical) sensor. Fallback mode is manual.

Change settings
The XML command is equivalent to the one above. Each channel can be set independently using a single command. Keep in mind that those settings are still volatile. They will be lost upon a system restart. To save it non-volatile, use the following command:

```
<analogout>
    <save>
        [<module> 1 .. 4 </module>]
    </save>
</analogout>
```

9.21. Mapping and Virtual Streams

User Offset

```
<mapper>
    <uoffset_set>
        <stream> phys1..phys15 | virt1..virt32 </stream>
        <value> v </value>
    </uoffset_set>
</mapper>
```

This command inserts a user offset in order that the corresponding stream output values will be v. Note that value v is in the same physical unit as the stream.
The following command removes the user offset.

```
<mapper>
    <uoffset_clr>
        <stream> phys1..phys15 | virt1..virt32 </stream>
    </uoffset_clr>
</mapper>
```

Virtual Streams
The following command is used to define virtual streams. Virtual streams are named virt1 to virt32. Virtual stream data can be used in expressions as DATA_VIR(1) to DATA_VIR(32). See also table 20 for all available data variables.

```
<mapper>
    <def_vstream>
        <virt1> expr1 </virt1>
        <virt2> expr2 </virt2>
        ...
        <virt32> expr32 </virt32>
    </def_vstream>
</mapper>
```

The expressions of virtual streams (not to be confused with expressions in the sequencer program) are any arithmetic formulas in C notation, for example:

- 100.5
  (a constant).
- DATA_PHY(13)*1.5
  (1.5 times the physical stream number 13).
- sqrt(DATA_PHY(2))
  (square root of physical stream number 2).
  pow(sin(DATA_VIR(1)), 2)
  (second power of sine of virtual stream number 1).
- -1000*(SYNTH - DATA_PHY(11))
  (this example uses the desired value from the synthesizer, normally used by the controller).

The order in which virt1, virt2, etc. are defined is only important if virtual streams depend on each other. During execution virtual streams are evaluated in the order of definition. See the follow- ing example:

```
<virt14> DATA_VIR(14)+1 <virt14>
<virt6> DATA_VIR(14)+1 <virt6>
```

Virtual stream number 14 is incremented in each time step. If at a certain time stream 14 stands at 100 the update in the next time step will be:

1 DATA_VIR(14) = DATA_VIR(14) + 1 = 100 + 1 = 101
2 DATA_VIR(6) = DATA_VIR(14) + 1 = 101 + 1 = 102

However, if the definition is done in the opposite order, the update will result as follows.
<virt6> DATA_VIR(14)+1 <virt6>
<virt14> DATA_VIR(14)+1 <virt14>

1 DATA_VIR(6) = DATA_VIR(14) + 1 = 100 + 1 = 101
2 DATA_VIR(14) = DATA_VIR(14) + 1 = 100 + 1 = 101

For more sophisticated calculations, there are some further expressions:

- VAR(n) Access to sequencer variables (see Table 20 for details).
- _AC_ON_ Is 1 when any wave generator is running, otherwise 0.
- L_AND Logic AND operator.
- B_AND Bitwise AND operator.
- ADR(x) C-style address operator.

The following command removes all virtual streams.
<mapper>
<clr_vstream> </clr_vstream>
</mapper>

Calculated Values
The following command is used to define calculated values. Calculated values are named calc1 to calc16. Calculated values can be used in expressions as DATA_CALC(1) to DATA_CALC(16). See also table 20 for further information. The difference between virtual streams and calculated values is the update rate. Virtual data streams are updated in every sampling interval and hence consume a considerable amount of computation power. Calculated values are more static. They are updated on a much lower rate. Actually the update rate is 1/100 times the one of the virtual data stream. Therefor calculated values consume consider- ably less computation power. Otherwise calculated values can be used exactly the same way as vir- tual data streams.

```
<mapper>
    <def_cstream>
        <calc1> expr1 </calc1>
        <calc2> expr2 </calc2>
        ...
        <calc16> expr16 </calc16>
```

```
        </def_cstream>
    </mapper>
```

Expressions are exactly the same as with virtual data streams. The following command removes all calculated values.

```
    <mapper>
        <clr_cstream> </clr_cstream>
    </mapper>
```

External Data Streams

Each controller can import data streams from other controllers in a multi-controller setup. He can also export local data streams to other controllers within the system. External data streams, if imported, are available on the streams ext1to ext16(see table 20).

The following commands exist to import and export data streams.

```
    <mapper>
        <export>
            <stream> ... </stream>
            <slot> ... </slot>
        </export>
    </mapper>

    <mapper>
        <import>
            <device> ... </device>
            <slot> ... </slot>
            <stream> ... </stream>
        </import>
    </mapper>
```

In the export command <stream> is a valid stream name from table 20. <slot> is a number between 1 and 31 and is something like an outgoing mailbox number. At import the parameter <device> depicts the slave number (see table 3) of the sender. <slot> is the slot number the sender used to export the required stream. And <stream> is one of ext1to ext16, see table 20.

Here is an example.

```
    // Transmitter, slave no. 5:
    <mapper>
        <export>
            <stream> synth </stream>
            <slot> 3 </slot>
        </export>
    </mapper>
```

```
// Receiver, slave no. 2:
<mapper>
    <import>
        <device> 5 </device>
        <slot> 3 </slot>
        <stream> ext8 </stream>
    </import>
</mapper>
```

In the previous example the synthesizer output stream of slave number 5 will be available on slave number 2 on the external data stream ext8.
All import and export settings can be removed by the following command.

```
<mapper>
    <clr_import> </clr_import>
</mapper>

<mapper>
    <clr_export> </clr_export>
</mapper>
```

Drag Indicators
Reset positive and negative drag indicators:

```
<mapper>
    <clr_drag>
        <pdrag> d_list </pdrag>
        <ndrag> d_list </ndrag>
    </clr_drag>
</mapper>
```

d_list is a semicolon separated list of stream names from table 20.

Push-Pull mode switching and configuration
Request the current configuration:

```
<mapper>
    <pp_config_get> </pp_config_get>
</mapper>
```

The controller will respond with a telegram containing all physical stream and valve polarities for both, push and pull mode:

```
<mapper>
```

```
<pp_config_get>
    <strpol_physX_push> 1 or -1 </...>
    <strpol_physX_pull> 1 or -1 </...>
    ...
    with X=1..13 (for all physical streams)
    ...
    <valvpol_physX_push> 1 or -1 </...>
    <valvpol_physX_pull> 1 or -< </...>
    ...
    with X=1..13 (for all physical controlling streams)
    ...
    <valvpol_virtX_push> 1 or -1 </...>
    <valvpol_virtX_pull> 1 or -< </...>
    ...
    with X=1..32 (for all virtual controlling streams)
    ...
    <valvpol_extX_push> 1 or -1 </...>
    <valvpol_extX_pull> 1 or -< </...>
    ...
    with X=1..16 (for all external controlling streams)
    ...
</pp_config_get>
</mapper>
```

Set a new configuration:

```
<mapper>
    <pp_config>
        ...Parameters according to command above...
    </pp_config>
    </mapper>
```

To set a new configuration, it's fine to use a subset of the parameters from above. Actually, all param- eters which are not to be changed can be omitted. The new configuration will apply immediately. Be aware that changing the push-pull configuration is allowed only in standby mode.

To save the configuration non-volatile on the controller, use the following command:

```
<mapper>
    <pp_config_save> </pp_config_save>
</mapper>
```

To switch between pull and push mode, the command below is provided:

```
<mapper>
    <pp_set>
        <mode> 'pull' or 'push' </mode>
    </pp_set>
```

</mapper>

This command is permitted allways except program mode. However, only the master can be advised to change push-pull mode. Slaves will use the same mode as the master. The last active mode setting is memorized on the master and resumed on the next startup.

The current active mode can be requested with:

```
<mapper>
    <pp_get> </pp_get>
</mapper>
```

The controller will respond according to the pp_set command.

Trimming trigger thresholds

Trigger thresholds are used within a sequencer program for flow control and/or limit supervision. These thresholds are usually fixed values, which are given within the sequencer program. However, if they are to be changed at runtime of the sequencer program, the trigger-trim function can be used.

Before trimming can occur, it must be configured (setup):

```
<mapper>
    <trigger_trim_set>
        <name> t_name </name>
        <step> st </step>
        <min> mi </min>
        <max> ma </max>
    </trigger_trim_set>
</mapper>
```

Parameter t_name is a trigger name from table 21. However, only triggers which have a threshold can be trimmed of course. Parameters st, mi and ma are real values. step means the stepsize per increment. That is the value that will be added to or subtracted from the trigger threshold for each increment. min and max are lower and upper limits of the trimmable range. Hence the threshold value will remain within these limits.

Once, the trimming has been set up, the threshold can be changed by the following command:

```
<mapper>
    <trigger_trim>
        [<inc> N </inc>]
    </trigger_trim>
</mapper>
```

The parameter inc can be omitted. In that case N defaults to 1.

Every time a threshold has been changed, an event message will be sent to the host. See chapter 5.4 (group 4 communication) and table 4 (event codes) for details.

9.22. Manifold control

Configuration
Die Konfiguration der Zapfstellensteuerung ist als XML-File gespeichert und wird beim Aufstarten automatisch eingelesen. Sie hat folgenden Aufbau:

```
<manifold>
    <config>
        <active> no(30) | yes </active>
        <delay_poweron_psens> t </...
        <delay_poweron_enable> t </...
        <delay_poweroff> t </...
        <delay_setupoff> t </...
        <delay_pumpoff> t </...
        <delay_poweron_setup> t </...
        <delay_sys_pressure> t </...
        <delay_out_pressure> t </...
    </config>
</manifold>
```

t: Zeit in Sekunden(muss > = 0 sein, für zulässige Bereiche siehe Tabelle 7). Die Konfiguration kann durch folgendes Kommando abgefragt werden:

```
<manifold>
    <config_get> </config_get>
</manifold>
```

Der Controller antwortet mit obigem Parametersatz.
Wurde die Konfiguration überschrieben (nicht über Webinterface), kann sie durch folgendes Kom- mando dauerhaft auf dem Disk gespeichert werden:

```
<manifold>
    <config_save> </config_save>
</manifold>
```

Bedeutung der Parameter:

30. Wenn active=no gesetzt ist, wird kein Ventil aktiviert und kein Druck überwacht.

Parameter	Bedeutung	Wert
active	Zapfstellensteuerung aktiv?	no oder yes
delay_poweron_psens	Verzögerung Leistung ein nach Pressostat Ausgang ok	float
delay_poweron_enable	Verzögerung Leistung ein nach Regelung ein (wenn Pressostat Ausgang nicht anspricht)	float
delay_poweroff	Verzögerung Leistung aus nach Regelung aus	float
delay_setupoff	Verzögerung Einrichten aus nach Regelung ein oder nach Einrichten ein	float
delay_pumpoff	Verzögerung Leckölpumpe aus nach Rege- lung aus	float
delay_poweron_setup	Verzögerung Leistung ein nach Einrichten aus	float
delay_sys_pressure	Verzögerung nach Abfall Systemdruck	float
delay_out_pressure	Verzögerung nach Abfall Zapfstellendruck	float

Table 27: Parameter der Zapfstellenkonfiguration

Spülen
Beim Spülen wird die Ueberwachung der Drucksensoren deaktiviert.

```
<manifold>
    <set>
        <flush> on | off </flush>
    </set>
</manifold>
```

Ob Spülen aktiv ist kann durch folgendes Kommando abgefragt werden:

```
<manifold>
    <get> </get>
</manifold>
```

9.23. Traversensteuerung

Configuration
Die Konfiguration der Traversensteuerung ist als XML-File gespeichert und wird beim Aufstarten automatisch eingelesen. Sie hat folgenden Aufbau:

```
<traversectrl>
    <config>
        <active> no(31) | yes </active>
        <prefill_time> t </prefill_time>
        <delay_move> t </delay_move>
    </config>
</traversectrl>
```

t: Zeit in Sekunden(muss >= 0 sein, für zulässige Bereiche siehe Tabelle 8). Die Konfiguration kann durch folgendes Kommando abgefragt werden:

```
<traversectrl>
    <config_get> </config_get>
</traversectrl>
```

Wurde die Konfiguration überschrieben (nicht über Webinterface), kann sie durch folgendes Kom- mando dauerhaft auf dem Disk gespeichert werden:

```
<traversectrl>
    <config_save> </config_save>
</traversectrl>
```

Bedeutung der Parameter:

Parameter	Bedeutung	Wert
active	Neue Traversensteuerung aktiv?	no oder yes
prefill_time	Zeitdauer für Vorfüllen bevor die Traverse entspannt wird	float
delay_move	Verzögerung nach Entspannung, bevor Auf oder Ab aktiv wird.	float

Table 28: Parameter der Traversensteuerungskonfiguration

Steuerung
Das Verstellen der Traverse kann einerseits über die entsprechenden Tasten auf der Fernsteuerung ausgelöst werden oder über folgendes XML-Telegramm:

30. Wenn active=no gesetzt ist, so gilt die alte Funktionalität: Bei Tastendruck wird der zugewiesene digitale Ausgang aktiviert (Rückwärtskompatibilität).

```
<traversectrl>
    <move>
        <direction> up | down | stop </active>
    </move>
</traversectrl>
```

9.24. Spannkopfsteuerung

Configuration
Die Konfiguration der Spannkopfsteuerung ist als XML-File gespeichert und wird beim Aufstarten automatisch eingelesen. Sie hat folgenden Aufbau:

```
<clampctrl>
    <config>
        <active> no(32) | yes </active>
        <Upper_clamp_autotime> t </Upper_clamp_autotime>
        <Lower_clamp_autotime> t </Lower_clamp_autotime>
        <force_control>off | manual | auto</force_control>
        <force_display> no | yes </force_display>
        <force_adjust> no | yes </force_adjust>
        <force_src> channel </force_src>
        <force_default> val </force_default>
        <dither_freq> val </dither_freq>
        <dither_ampl> val </dither_ampl>
        <current_max> val </current_max>
    </config>
</clampctrl>
```

t: Zeit in Sekunden(muss > = 0 sein, für zulässige Bereiche siehe Tabelle 9). Die Konfiguration kann durch folgendes Kommando abgefragt werden:

```
<clampctrl>
    <config_get> </config_get>
</clampctrl>
```

Wurde die Konfiguration überschrieben (nicht über Webinterface), kann sie durch folgendes Kom- mando dauerhaft auf dem Disk gespeichert werden:

```
<clampctrl>
    <config_save> </config_save>
</clampctrl>
```

121. Wenn active=no gesetzt ist, so gilt die alte Funktionalität: Bei Tastendruck wird der zugewiesene digitale Ausgang aktiviert (Rückwärtskompatibilität).

Bedeutung der Parameter:

Parameter	Bedeutung	Wert
active	Neue Spannkopfsteuerung aktiv?	no oder yes
Upper_clamp_autoti me	Zeitdauer für Aktivierung des Oeffnen-Ventils	float
Lower_clamp_autoti me	Zeitdauer für Aktivierung des Oeffnen-Ventils	float
force_control	Modus für Klemmenkraftsteuerung	no, manual, auto
force_display	Anzeige der Klemmenkraftwerte (auf Fernsteuerung)	no, yes
force_adjust	Anzeige des Klemmenkraft-Einstellschiebers (auf Fernsteuerung)	no, yes
force_src	Messkanal für die Klemmenkraft	none oder physX(1)
force_default	Default-Wert für die Klemmenkraft in %	float
dither_freq	Ditherfrequenz (40 bis 200 Hz)	float
dither_ampl	Ditheramplitude (0 bis 50%)	float
current_max	Maximaler Ausgangsstrom, welcher 100% entspricht (0 bis 1A)	float

Table 29: Parameter der Spannkopfsteuerungskonfiguration
1. siehe Tabelle 20

Steuerung
Die Steuerung der Spannköpfe kann einerseits über die entsprechenden Tasten auf der Fernsteuerung erfolgen oder über folgendes XML-Telegramm:

```
<clampctrl>
    <move>
        <upper | lower> open | close | stop </...>
    </move>
</clampctrl>
```

Anmerkung: Der Stop-Parameter wird nicht zwingend benötigt. Beim Schliessen geht die Steuerung automatisch in den Selbsthaltemodus, sobald der entsprechende Druckschalter anspricht. Ein Stop-

Befehl ist dann wirkungslos. Beim Oeffnen wird das entsprechende Ventil nach Ablauf der eingest- ellten Zeit (siehe obige Tabelle) automatisch stromlos.

10. Literature

[1] Stettbacher Signal Processing (2007). "W+B Maschinensteuerung - Konzept für Ethernet-Kommunikation", Walter+Bai Prüfmaschinen AG, CH-8224 Löhningen.

[2] Stettbacher Signal Processing (2011). "Der erweiterte PIDV-Regler - Regler für elektromechanische Prüfmaschinen, Version 0.97", Walter+Bai Prüfmaschinen AG, CH-8224 Löhnin- gen.