

Undulator Control at BESSY

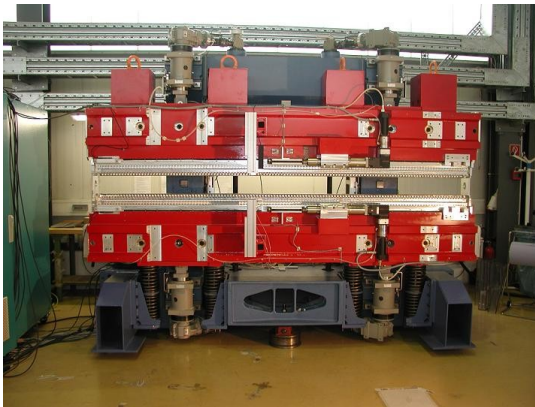
Overview

- Undulator Components
- Model for communication
- Ways to communicate
- Standard movement
- Simultaneous movement of gap and shift
- Dynamic speed control
- Documentation

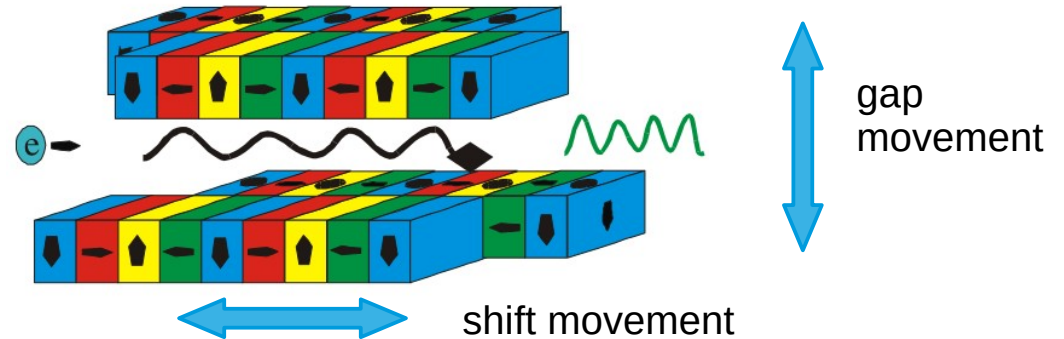
Undulator components, shown at UE49

- 2 split bars of permanent magnets above and below vacuum chamber
- Motors control vertical movement towards chamber: gap drive
- Motors control horizontal movement of split bars: shift drive

UE49

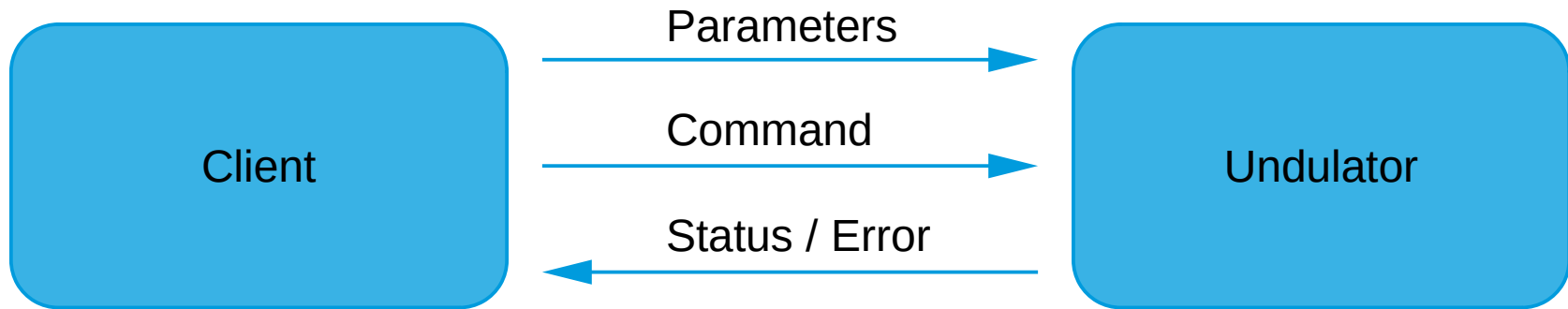


magnet layout



Model for communication

- Command, e.g. “START”, “STOP”
- Parameters, e.g. “destination gap”
- Status and Error, e.g. “run”, “stop”



Ways to communicate

- Channel access
 - Easy to use, many CA clients exist
 - Only a network connection is needed
 - Access control via channel access gateway
- CAN Bus
 - CAN Interface needed at the client
 - CAN bus cable to undulator needed
 - Better real-time capabilities

Standard movement: ID user panel

The image shows a software interface for the Undulator UE49. The interface is divided into two main sections: 'Destination-Gap' and 'Destination-Shift'. At the top, there are controls for 'ID control' (set to 'local'), 'couple gap/shift' (checkbox), and 'dynamic velocity' (checkbox). The 'shift mode' is set to 'antiparallel+'. The '1st harmonic (at shift = 0 mm)' is 560.022 eV.

Destination-Gap Section:

- Destination-Gap:** 19.210 [mm]
- Command:** START
- exec** (green button)
- STOP** (red button)
- gap velocity:** 0.1397
- delta gap:** 0.0005
- return position:** 21.000
- actual gap [mm]:** 100.001
- Status:** STOP
- Error:** None

Destination-Shift Section:

- Destination-Shift:** 21.378 [mm]
- Command:** START
- exec** (green button)
- shift velocity:** 0.4200
- delta shift:** 0.0005
- actual shift position [mm]:**
 - Parallel: -0.000
 - AntiParallel: 21.378
- Status:** STOP
- Error:** None

Annotations:

- Drive mode for shift:** Points to the 'ID control' dropdown.
- Execute command for gap:** Points to the 'exec' button in the gap section.
- Gap velocity:** Points to the 'gap velocity' field.
- Execute command for shift:** Points to the 'exec' button in the shift section.
- Shift velocity:** Points to the 'shift velocity' field.
- Destination-Gap:** Points to the 'Destination-Gap' input field.
- Gap Command:** Points to the 'START' button in the gap section.
- Destination-Shift:** Points to the 'Destination-Shift' input field.
- Shift Command:** Points to the 'START' button in the shift section.

Standard movement: principle steps

Gap- and shift drive work like independent devices, you have to do the following for the gap- and the shift drive:

- Set movement parameters:
 - Destination
 - Velocity (optional)
 - Drive mode (optional)
- Give Command “RUN”
- Monitor run-counter, status and error during movement

Standard movement: end of move detection

How to properly detect that a movement is finished

- The status usually is either „RUN“ or „STOP“
- Longer movements change the status from “STOP” to “RUN” and back again to “STOP”
- For very short movements, the status may remain „STOP“ all the time
- The run-counter is guaranteed to increase each time a movement is finished.

➔ If you must wait for a movement to finish, always monitor the run-counter

Simultaneous Movement of Gap and Shift

The command always goes to gap- and shift drive. Both drives start and stop moving at exactly the same time. These are the steps:

- Select “couple gap/shift”
- Set movement parameters
 - Destination for gap and shift
 - Velocity for gap and shift (optional)
 - Drive mode for shift (optional)
- Give Command “RUN” (to gap drive, starts shift, too)
- Monitor gap/shift run counter, status and error during movement

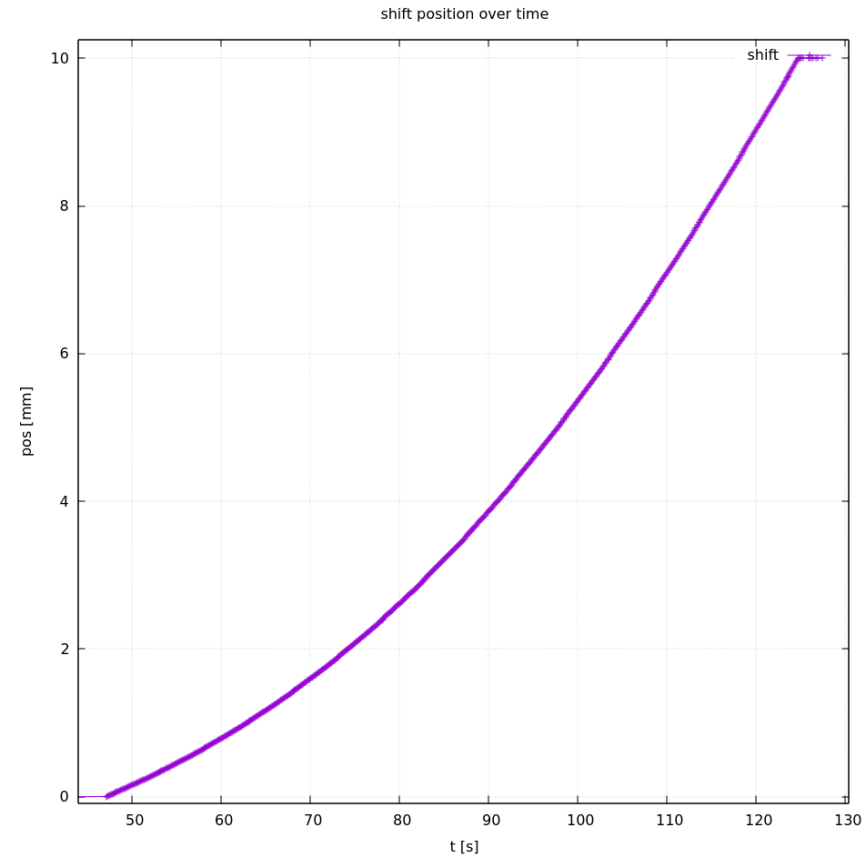
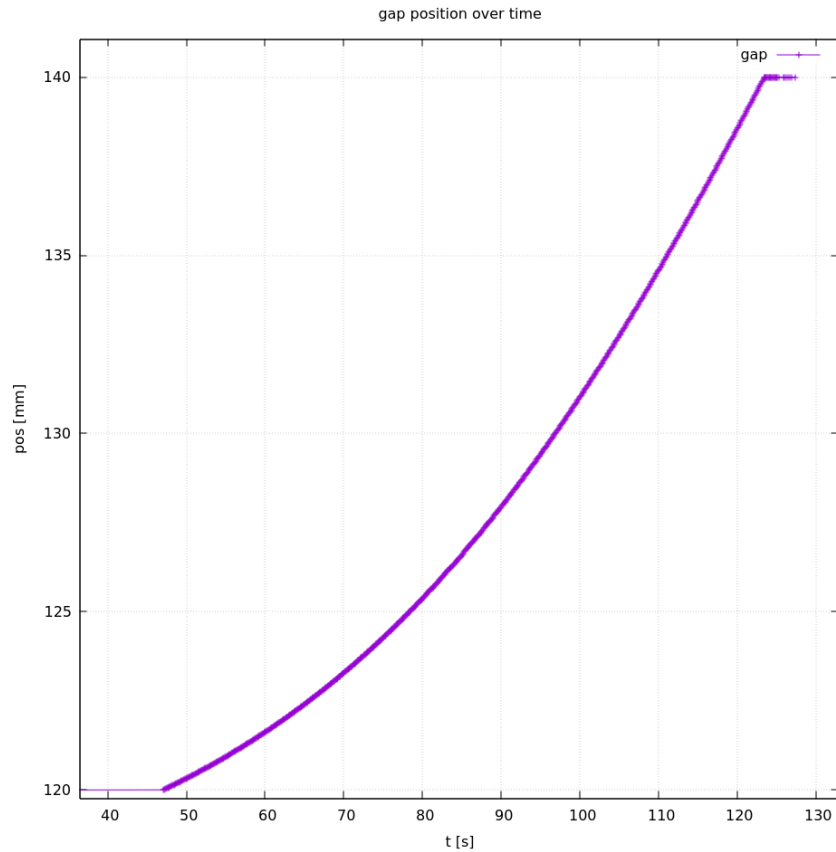
Dynamic speed control

- Gap and shift movement speed can be changed during movement from 0% to 100% with a resolution of 16 bits each.
- Available for Channel Access and CAN Bus
- Gap- and shift velocity guaranteed to change at the same time since they are transferred as a single value (Channel Access: waveform, CAN Bus: 32 bit data frame)

Dynamic speed control – test setup

- Speed setting with a python script using caproto library
- Set velocity dependent on current gap with a rate of about 10 Hz
- $v(x)$ is a quadratic function, taken from a table with linear interpolation
- Position is taken from Ndi[n]PmsPos records that are updated with 10 Hz with camonitor
- Current velocity is calculated from measured positions and time stamps
- Results plotted with gnuplot

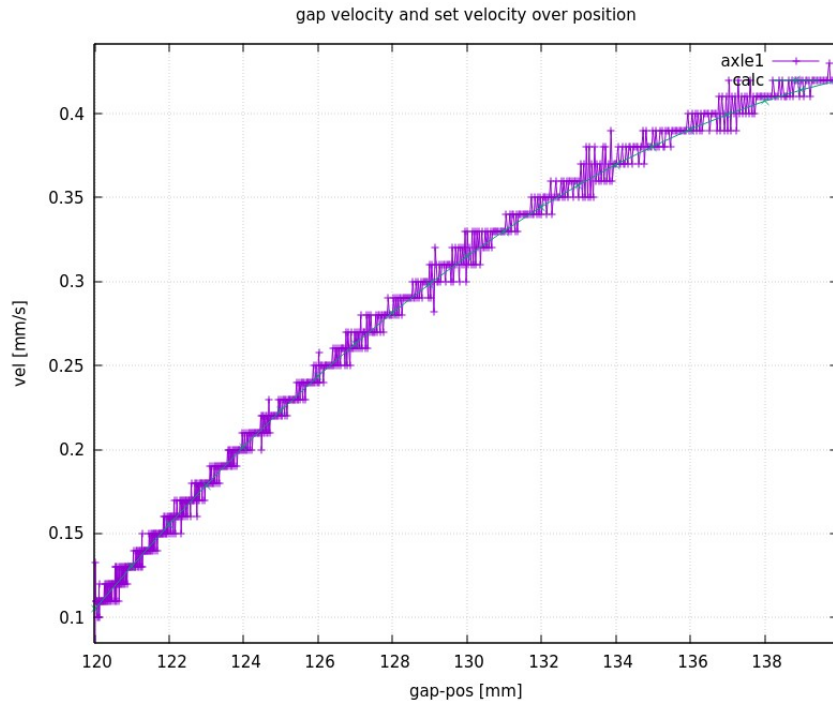
Dynamic speed control – test results



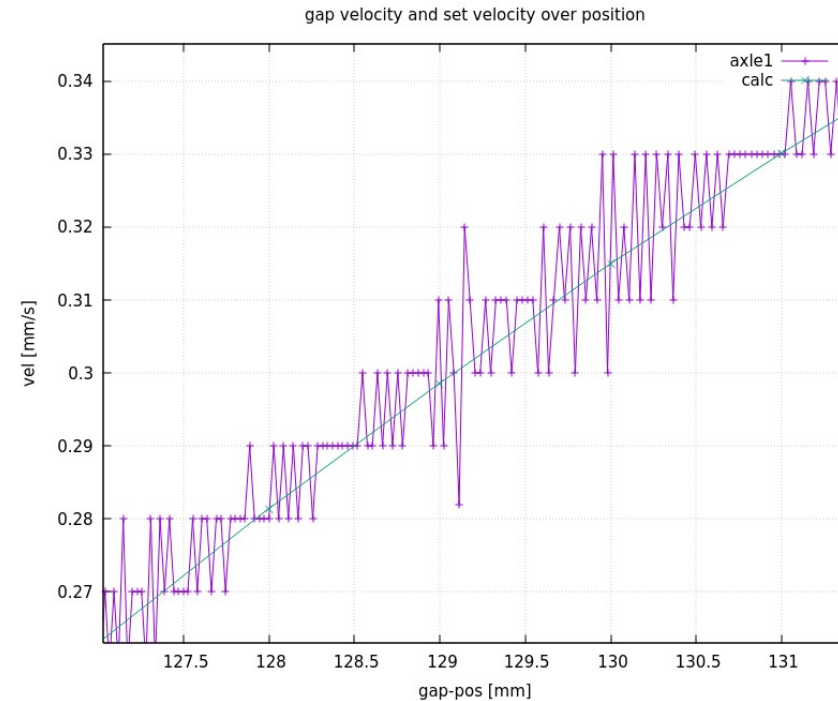
Dynamic speed control – test results

Measured velocity versus set-velocity

Full scan

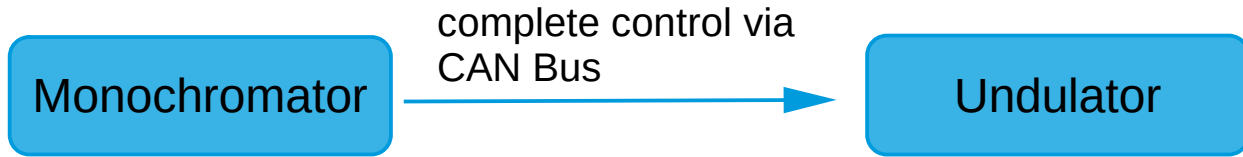


detail

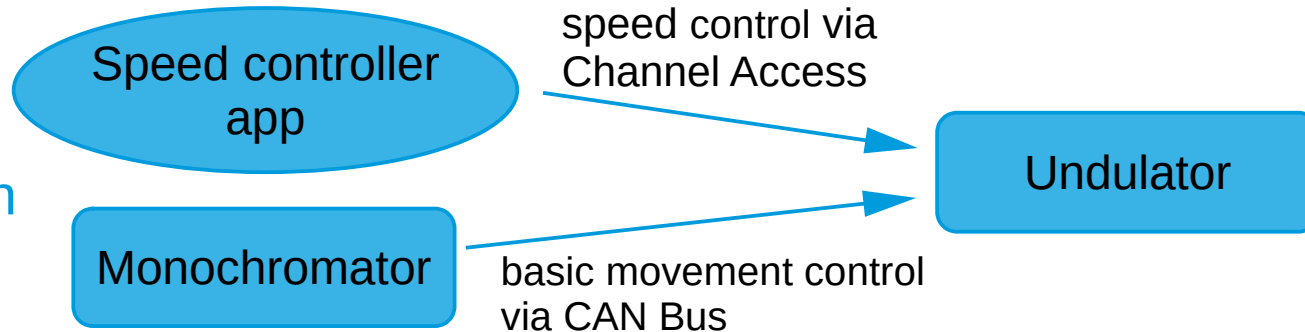


Dynamic speed: possible configurations

Speed control by monochromator



Speed control by external application



Speed control on insertion device



Documentation

- **Main web site**
<https://www-csr.bessy.de/control/idcp-documentation/>
- **Common user interface**
https://www-csr.bessy.de/control/idcp-documentation/idcp_user_interface.html
- **CAN Bus Interface**
https://www-csr.bessy.de/control/idcp-documentation/wup_can.html
- **Dynamic Speed**
https://www-csr.bessy.de/control/idcp-documentation/idcp_user_interface.html#dynamic-speed