Helmholtz Zentrum Berlin

EPICS INTEGRATION FOR RAPID CONTROL PROTOTYPING HARDWARE FROM SPEEDGOAT

THPDP013

L. Rossa, M. Brendike   Helmholtz-Zentrum Berlin für Materialien und Energie GmbH, 14109 Berlin, Germany

ABSTRACT: To exploit fourth generation Synchrotron Sources new beamline instrumentation is increasingly developed with a mechatronics approach. [1,2,3] Implementing this raises the need for Rapid Control Prototyping (RCP) and Hardware-In-the-Loop (HIL) simulations. To integrate RCP and HIL systems into beamline operation we developed an interface from a Speedgoat real-time performance machine to EPICS. The interface was developed to be flexible and simple to use. The Simulink software developer uses dedicated Simulink-blocks to export model information and real-time data into structured UDP Ethernet frames. The corresponding EPICS IOC listens to the UDP frames and auto-generates a corresponding database file to fit the data-stream from the Simulink model. The EPICS IOC can run on either a beamline measurement PC or, to keep things spatially close on a mini PC (such as a Raspberry Pi) attached to the Speedgoat machine.

REQUIREMENTS

- RCP or HIL system should be integrable into a beamline via plug-and-play
- the developer behind the RCP or HIL system is not an EPICS expert
- the developer should not struggle to keep EPICS in sync with the RCP or HIL system
- the beamline scientist or user is not an expert in the RCP or HIL program
- the full flexibility of the RCP and HIL system should be maintained

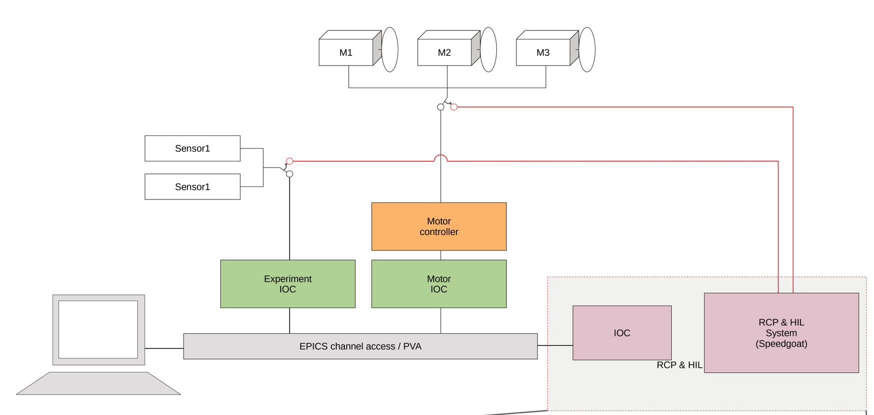

Figure 1: HIL & RCP system at an example beamline

IMPLEMENTATION

- UDP used for communication
- Speedgoat PC sends regular model/program information
- EPICS IOC auto-generates *.db file and configures itself
- Changes to a Speedgoat variable value result in an EPICS PV change
- Changes to an EPICS PV or field result in an UDP dataframe to Speedgoat, which handles the change
- Speedgoat is reachable via channel access
- Support for EPICS motor record

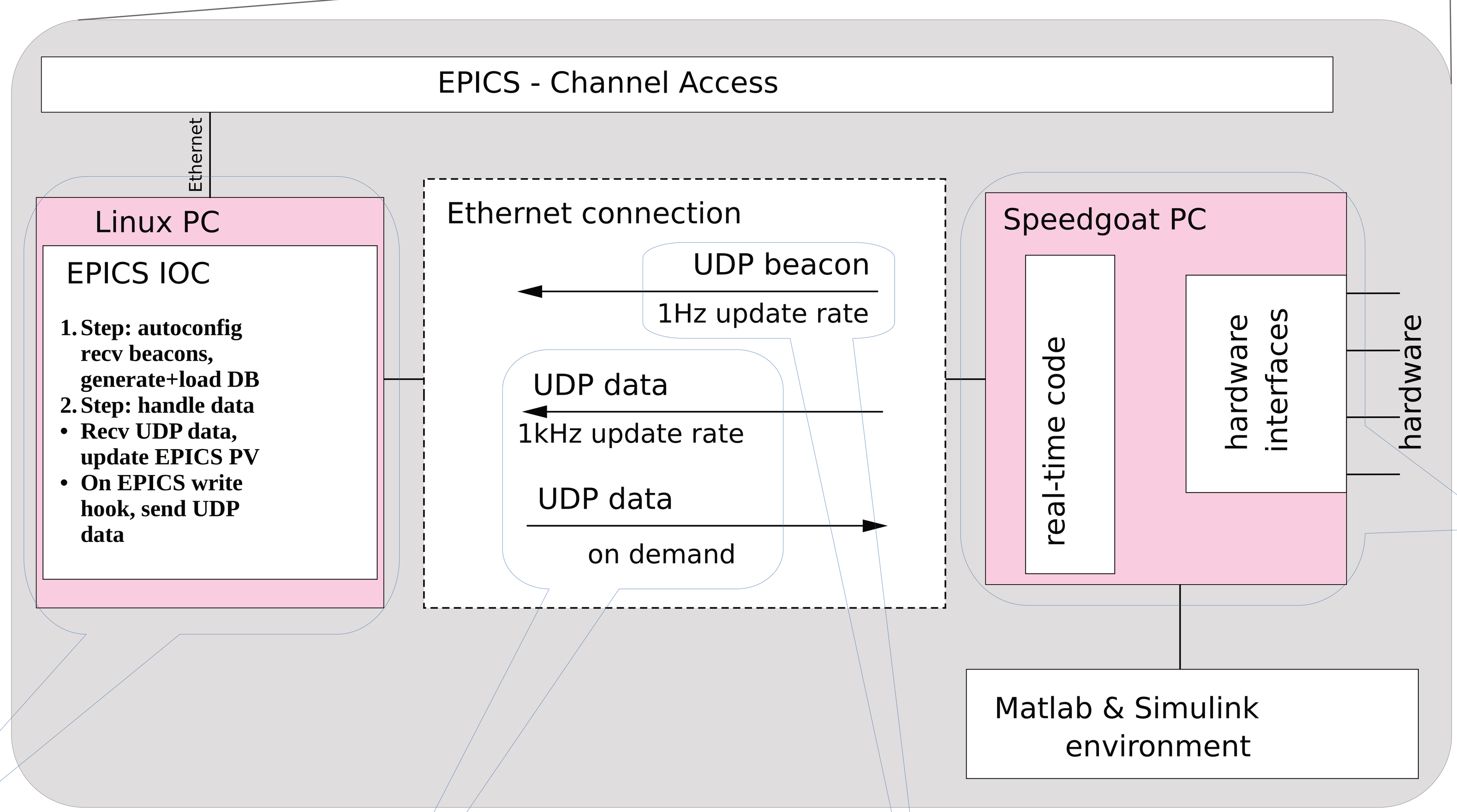

Figure 3: Communication between EPICS IOC and Speedgoat Hardware

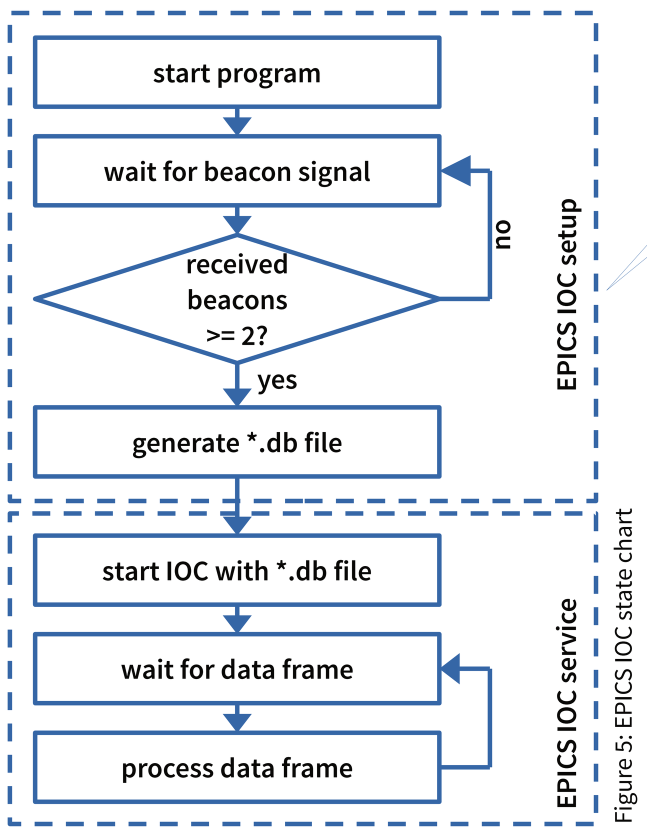

Figure 5: EPICS IOC state chart

```
# object-id 2, data-type 5/int32, object-type 20/motor-long
record(motor, "SPEEDGOAT:Epics_motor1") {
..field(DTYP, "asynMotor")
..field(OUT, "@asyn(IP,1,1)")
..…
}

# object-id 1, data-type 5/int32, object-type 40/encoder
record(longin, "SPEEDGOAT:epics_enc_pos") {
..field(DTYP, "asynInt32")
..field(INP, "@asyn(IP,0,1)SG_epics_enc_pos")
..…
}
```

Figure 6: Generated DB for EPICS IOC

EPICS IOC

- EPICS base R7.0.7
- Asyn R4-43
- Motor support R7-2-1

UDP data analog input (AI)
192.168.167.42 → 192.168.167.41 : 18065

	offset	name	type	example	hexdump
meta data	0…7	timestamp (s)	*double*	12.345	0x71 3d 0a d7 a3 b0 28 40
		optional meta data			
object info 1	8	object-id	*uint8*	1	0x01
	9	object-type	*uint8*	1 (AI)	0x01
	10	data type	*uint8*	5 (int32)	0x05
	11…12	value-count	*uint16*	1	0x01 0x00
	13…52	PV name	*string*	EPICS_ai1	0x45 50 49 43 53 5f 61 69 31 00…
	53…56	value	*int32*	42	0x2a 0x00 0x00 0x00
obj.2	57	object-id	*uint8*	0	0x00 (no more data)
	58…999	padding	*uint8*	0	0x00

UDP data analog output (AO)
192.168.167.41 → 192.168.167.42 : 18066

	offset	name	type	example	hexdump
meta data	0…7	timestamp (s)	*double*	13.456	0xb6 f3 fd d4 78 e9 2a 40
		optional meta data			
object info 1	8	object-id	*uint8*	7	0x07
	9	object-type	*uint8*	2 (AO)	0x02
	10	data type	*uint8*	5 (int32)	0x05
	11…12	value-count	*uint16*	1	0x01 0x00
	13…52	PV name	*string*	EPICS_ao3	0x45 50 49 43 53 5f 61 6f 33 00…
	53…56	value	*int32*	42000	0xa4 0x10 0x00 0x00
obj.2	57	object-id	*uint8*	0	0x00 (no more data)
	58…999	padding	*uint8*	0	0x00

Figure 7: Examples of exchanged UDP data packets

UDP communication

- UDP used for simplicity
- Currently UDP packages limited to 1000 bytes to avoid fragmentation

UDP beacon
192.168.167.42 → 255.255.255.255 : 18064

	offset	name	type	example / hexdump	description
model info	0…7	timestamp (s)	*double*	7.314 0xa8 c6 4b 37 89 41 1d 40	time in seconds since program start
	8	version	*uint8*	1 0x01	always 1 to fix format
	9…48	application	*string*	sgDemo 0x73 67 44 65 6d 6f 00…	program name filled with NUL or SPACE (fixed length 40 bytes)
object info 1	49	object-id	*uint8*	1 0x01	object ID of 1st. object
	50	object-type	*uint8*	1 (AI) 0x01	other object types possible: AI,AO,DI,DO,motor,…
	51	data type	*uint8*	5 (int32) 0x05	other data types possible: (un)signed int, float, …
	52…55	ip_addr_int	*uint32*	192.168.167.42 0x2a 0xa7 0xa8 0xc0	IPv4 address of Speedgoat machine
	56…57	udp_port_int	*uint16*	18065 0x91 0x46	Speedgoat UDP port
	58…61	ip_addr_ext	*uint32*	192.168.167.41 0x29 0xa7 0xa8 0xc0	IPv4 address of EPICS-PC
	62…63	udp_port_ext	*uint16*	18065 0x91 0x46	EPICS-PC UDP port
	64…103	PV name	*string*	EPICS_ai1 0x45 50 49 43 53 5f 61 69 31 00…	name of the EPICS PV (fixed length 40 bytes)
object info 2	104	object-id	*uint8*	7 0x07	object ID of 2nd. Object
	105	object-type	*uint8*	2 (AO) 0x02	other object types possible
	106	data type	*uint8*	5 (int32) 0x05	other data types possible
	107…110	ip_addr_int	*uint32*	192.168.167.42 0x2a 0xa7 0xa8 0xc0	IPv4 address of Speedgoat machine
	111…112	udp_port_int	*uint16*	18065 0x91 0x46	Speedgoat UDP port
	113…116	ip_addr_ext	*uint32*	192.168.167.41 0x29 0xa7 0xa8 0xc0	IPv4 address of EPICS-PC
	117…118	udp_port_ext	*uint16*	18065 0x91 0x46	EPICS-PC UDP port
	119…158	PV name	*string*	EPICS_ao3 0x45 50 49 43 53 5f 61 6f 33 00…	name of the EPICS PV (fixed length 40 bytes)
end of info	159	object-id	*uint8*	0 0x00	means no more objects (normally object ID of 3rd. Object)
	160…999	padding	*uint8*	0 0x00	

Figure 8: Example of UDP beacon packets

Simulink EPICS library

- Simulink library for EPICS objects is created
- Send and receive blocks for configuring the UDP connection

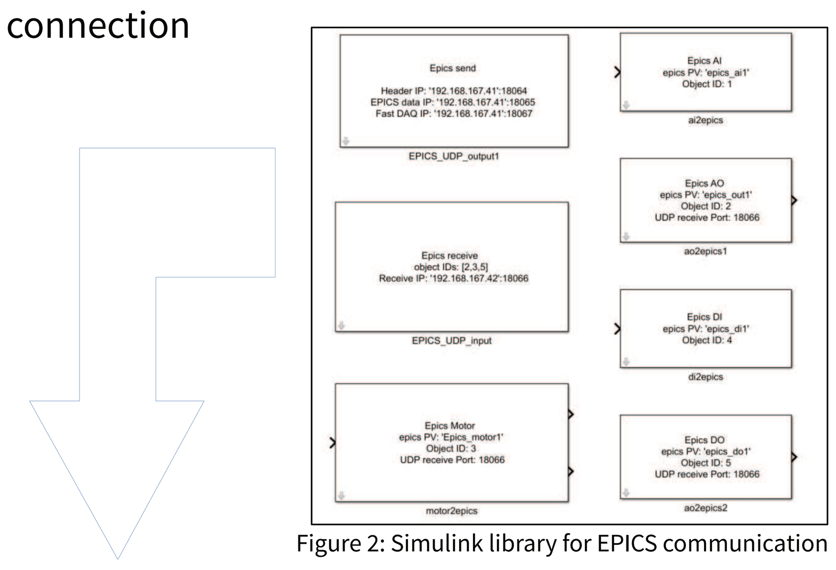

Figure 2: Simulink library for EPICS communication

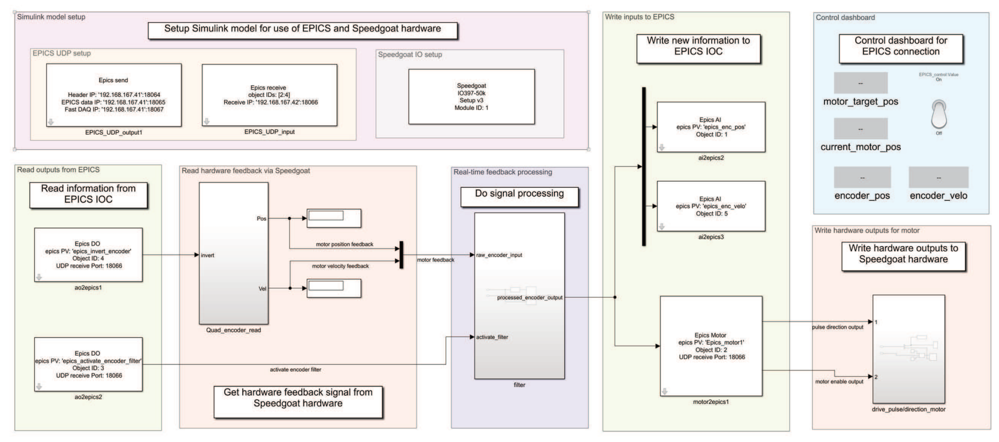

Figure 4: Simulink example model for RCP on a single stepper motor

Example setup

- Requirement was a feedback filter for a stepper motor with direct encoder feedback
- Setup hardware:
  - 200 steps per revolution stepper motor,
  - 4000 increments per revolution encoder.

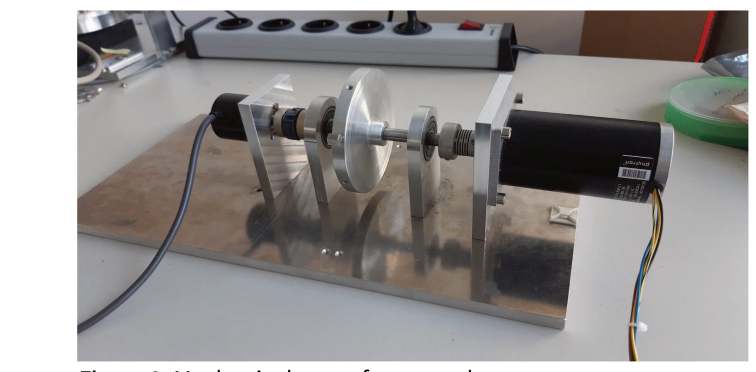

Figure 9: Mechanical setup for example setup

CONCLUSION

- An interface from Speedgoat hardware to EPICS was created
- Proof of principle shows that the interface works
- Full performance benchmark still needs to be done
- A full integration of a bigger system than the example setup will be the next step

REFERENCES

[1] R. R. Geraldes et al., "Mechatronics Concepts for the New High-Dynamics DCM for Sirius," in Proc. 9th Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation (MEDSI'16), Barcelona, Spain, 11-16 September, 2016, no. 9 in Mechanical Engineering Design of Synchrotron Radiation Equipment and Instrumentation Conference, (Geneva, Switzerland), pp. 44–47, JACoW, June 2017. https://doi.org/10.18429/JACoW-MEDSI2016-MOPE19

[2] M. Brendike et al., "ESRF-Double Crystal Monochromator Prototype - Control Concept" in Proc. 17th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS19), New York, NY, USA, 05-11 October 2019 , pp 776-780, JACoW, August 2020, https://doi.org/10.18429/JACoW-ICALEPCS2019-TUCPL05

[3] T. Dehaeze, J. Bonnefoy, and C.G.R.L. Collette, "Mechatronics Approach for the Development of a Nano-Active-Stabilization-System", in Proc. MEDSI'20, Chicago, IL, USA, Jul. 2021, pp. 93–98. doi:10.18429/JACoW-MEDSI2020-TUIO02

ACKNOWLEDGEMENTS

Special thanks to Roland Fleischhauer, Olaf Pawlizki, Rayk Horn, Markus Neu and David Kraft for helping with the hardware integration of the Speedgoat PC into the BESSY II electronics environment.

Thanks to the EPICS community
https://epics-controls.org/

MORE INFORMATION

Lutz Rossa <rossa@helmholtz-berlin.de>
+49-30-8062-42574
https://www.helmholtz-berlin.de/

Maxim Brendike <maxim.brendike@helmholtz-berlin.de>
+49-30-8062-18064
https://www.helmholtz-berlin.de/