Synchronised PS-SPS transfer with barrier buckets

LLRF22

Mihaly Vadai

Acknowledgements:
H. Damerau, M. Giovannozzi, A. Huschauer, A. Lasheen the Operations Team and SY-RF-FB

10-10-2022

Outline

Motivation

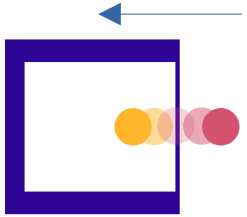
Synchronisation concept

Prototype hardware and firmware

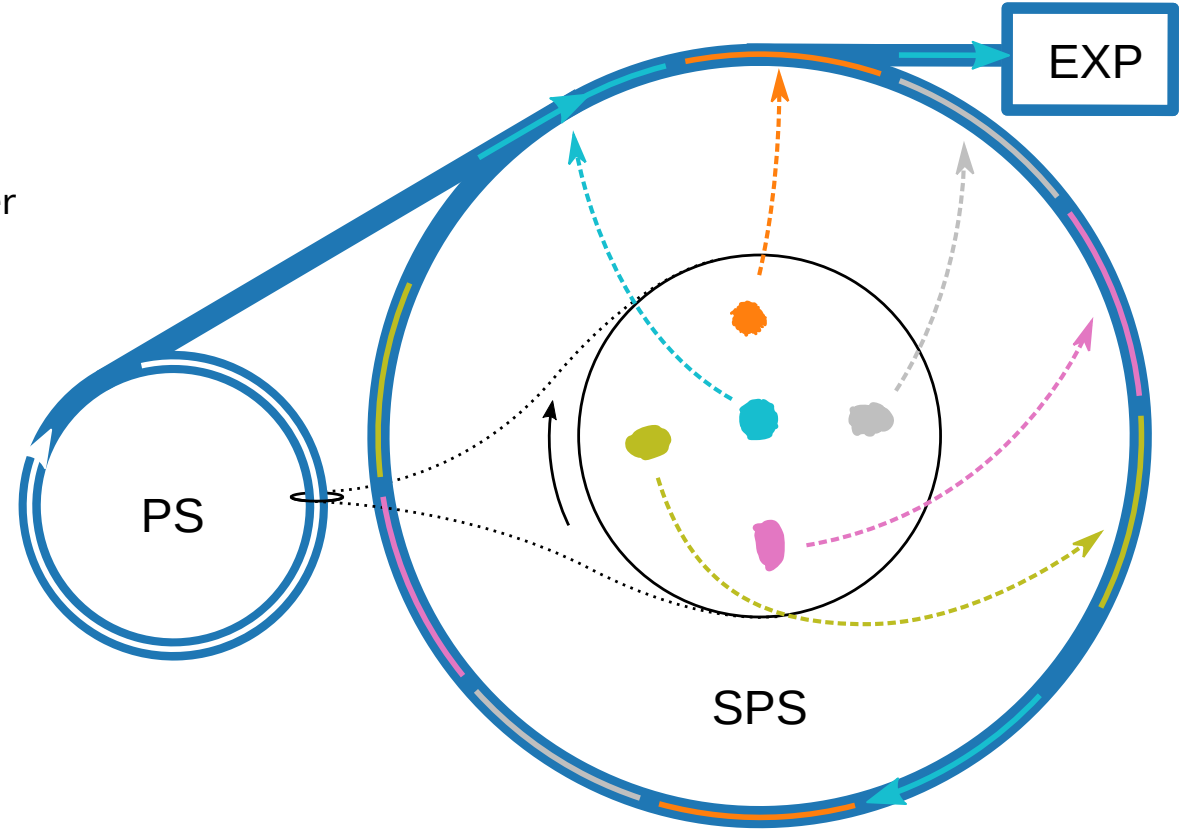
Setup of synchronisation with beam in barrier buckets

Motivation

- Beam loss reduction at 5 turn extraction of a coasting beam
- Losses occur during the rise time of the kicker on the dummy septum



- Making a longitudinal gap in the beam during kicker rise time with a barrier bucket reduces losses:
  https://iopscience.iop.org/article/10.1209/0295-5075/128/14002
- New: RF needs to be synchronous with SPS



RF = radio frequency, PS = Proton Synchrotron, SPS = Super Proton Synchrotron, EXP = experiment

Wide-band RF needed to make a gap

RF waveform and longitudinal phase space

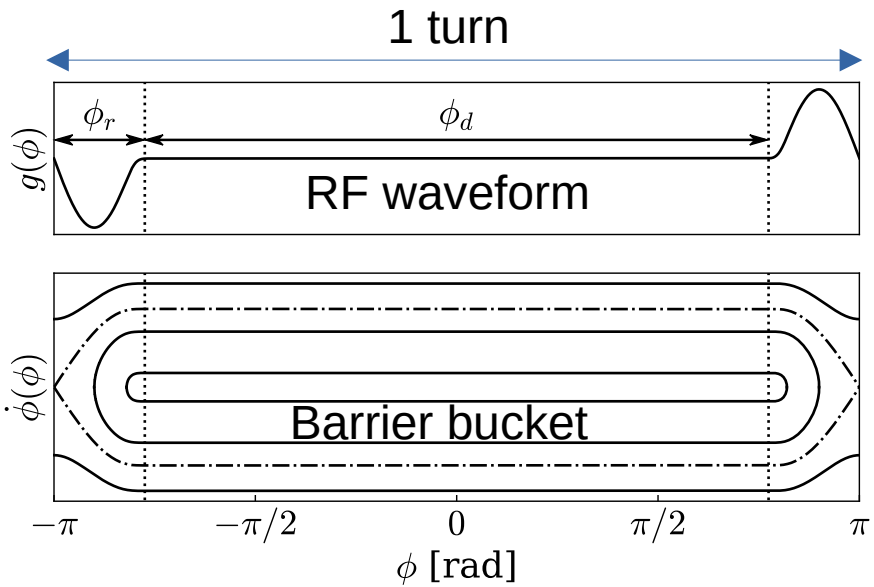


Diagram from M. Vadai et. al. Barrier bucket gymnastics and transversely split proton beams: Performance at the CERN Proton and Super Proton Synchrotrons (CC-BY 4.0)
https://link.aps.org/doi/10.1103/PhysRevAccelBeams.25.050101

Finemet wideband cavity in PS is suitable to try this scheme

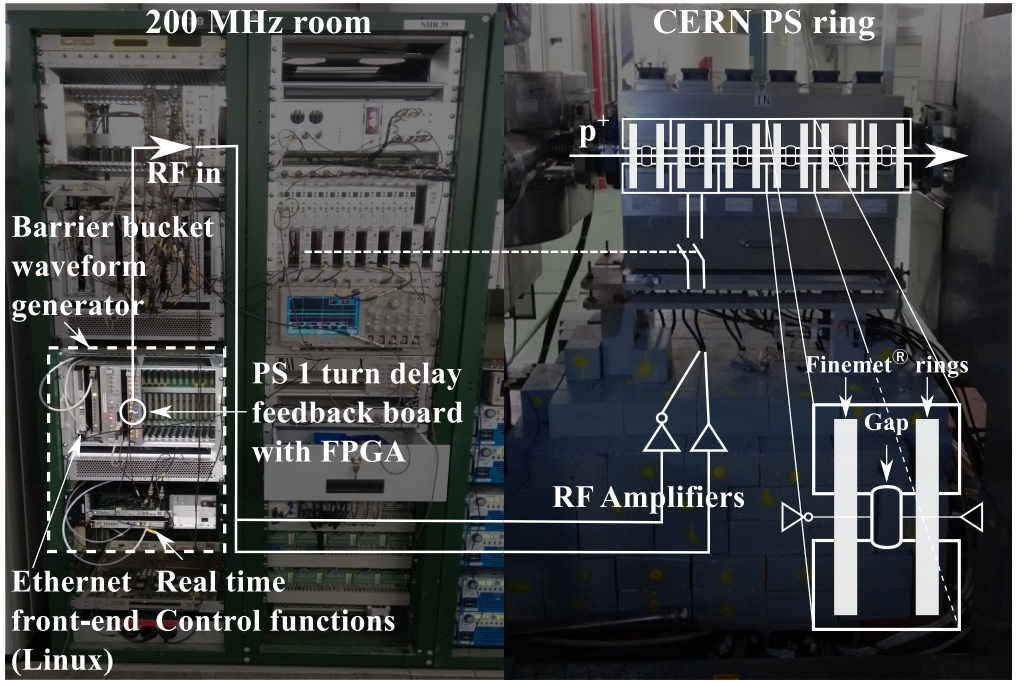


Diagram from M. Vadai, A. Alomainy, H. Damerau: Barrier Bucket Studies in the CERN PS (CC-BY 3.0)
https://doi.org/10.18429/JACoW-IPAC2019-MOPTS106
See also: https://doi.org/10.18429/JACoW-IPAC2019-MOPTS107

Conceptual choice for synchronization

Classical synchronization difficult:

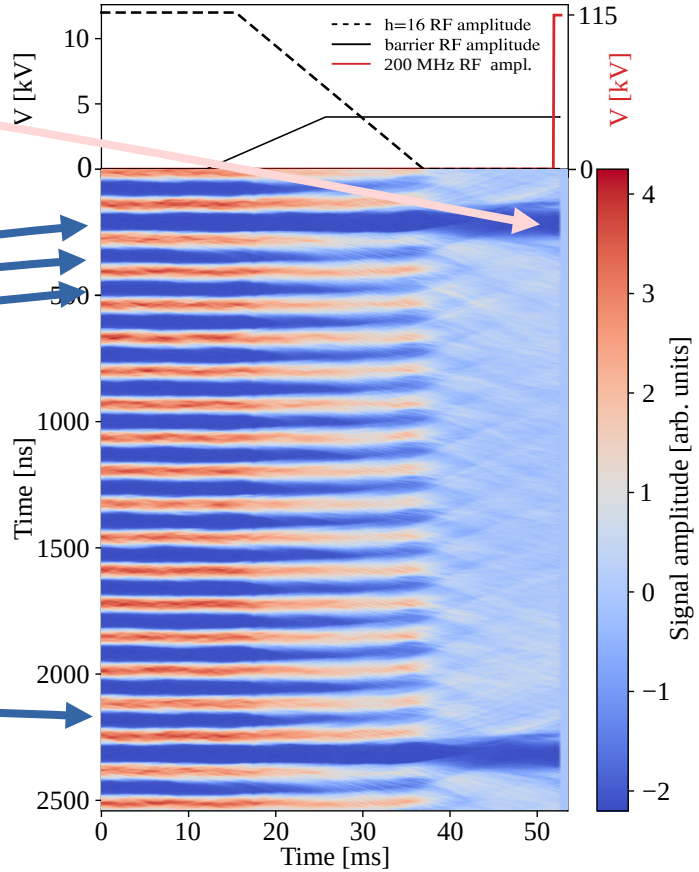
- Synchronization is changing the energy of the beam
- Transverse splitting needs low RF voltage and a centered beam
- *h*=1 synchronization can not be done at the end of the cycle easily

Idea for synchronization:

- Measure *h*=16 phase and apply open-loop frequency bump
- Once *h*=16 synchronous, find gap between the buckets which is *h*=1 synchronous
- Bring up the barrier at the SPS synchronous spot, while still keeping the RF voltage low
- Respect the fixed radial positions needed for the transverse splitting and SPS extraction

Longitudinal profiles at the end of flat top

- Gap generated by the barrier to be synchronous with PS extraction and SPS injection
- Use re-bucketing as a guide: 16 possible positions for barrier gap
- Half of an *h*=16 bucket change instead of *h*=1 → substantial time saving



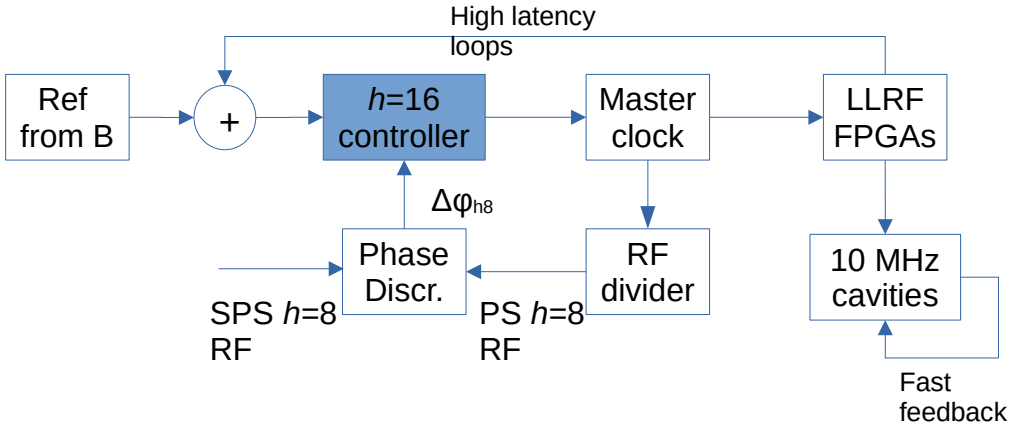
Synchronisation overview at fixed bending field

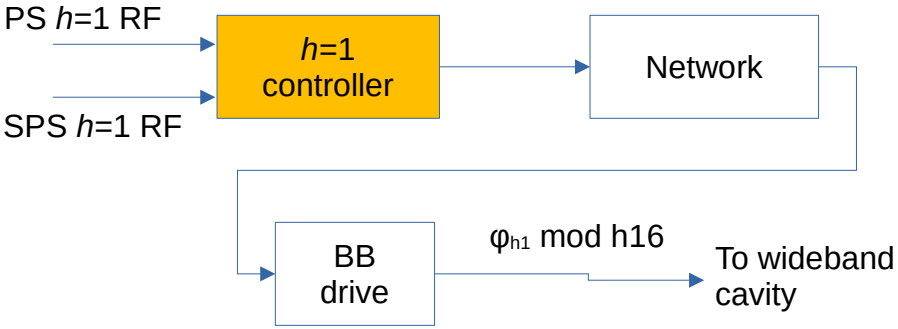
h=16
- smoothly switch off loops
- measure phase w.r.t. SPS

h=16
perform phase compensation to align gaps between RF buckets with SPS

h=1
measure phase w.r.t. SPS and send to the location of the Finemet drive

h=1
raise barrier between the correct *h*=16 buckets synchronously with SPS





Demonstrator boards needed for synchronization concept validation

Features needed

- Increase open loop revolution frequency word length from 16 to 32 bits
- Perform *h*=16, *h*=1phase measurements
- Calculate / play frequency programs that control the beam phase w.r.t. SPS

Constraints

- Latency < 1 µs in closed loop
- Jitter at extraction < few ns (beam-based)
- Simple, custom interface boards
- Preferably open toolchain
- High level code
  (only optimization at low-level)

Microcontroller on development board

STM32H735ZG (released Q4 2020):

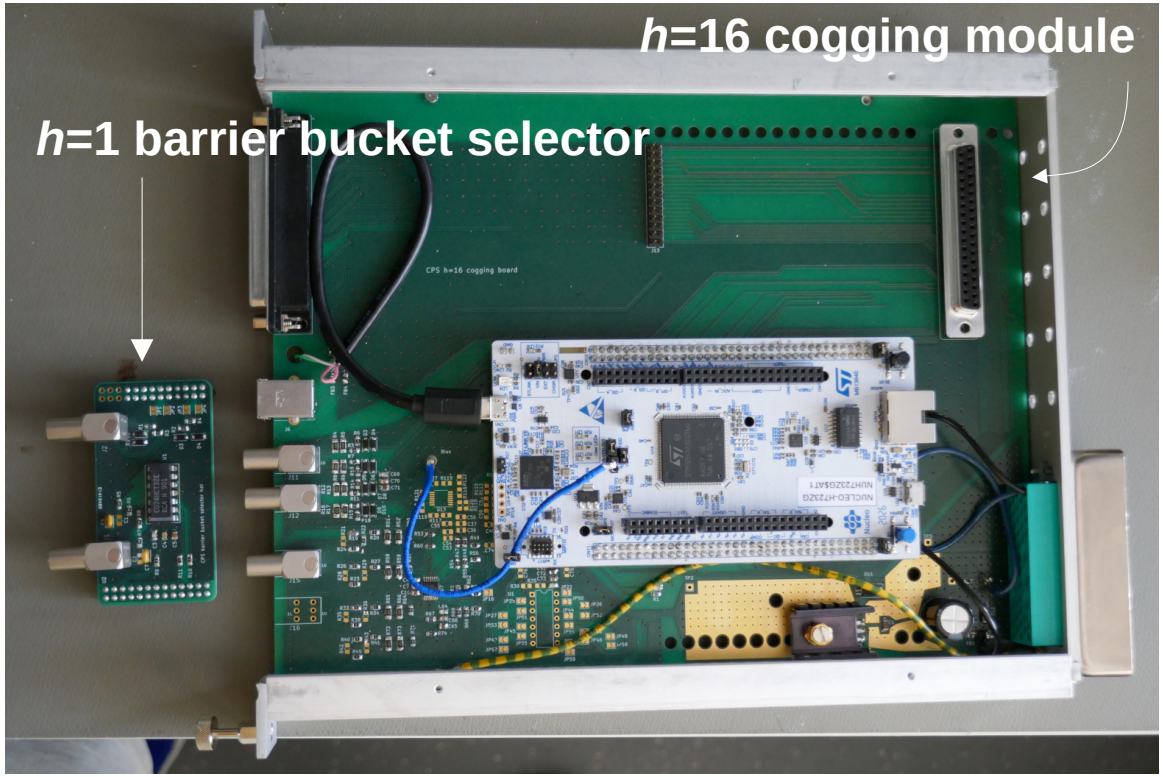
- Max. 550 MHz ARM Cortex M7 core, DP-FPU
- Self cal. ADCs, DAC, DMA, MDMA
- USB, FMC, OCTOSPI, FMAC, Cordic

Development board & toolchain:

- 144 pins, 1.8V, 3.3V, 5V regulators
- Additional USB debug, ethernet + PTP
- Open source compiler, code generator (simple init code, incl. ANN)

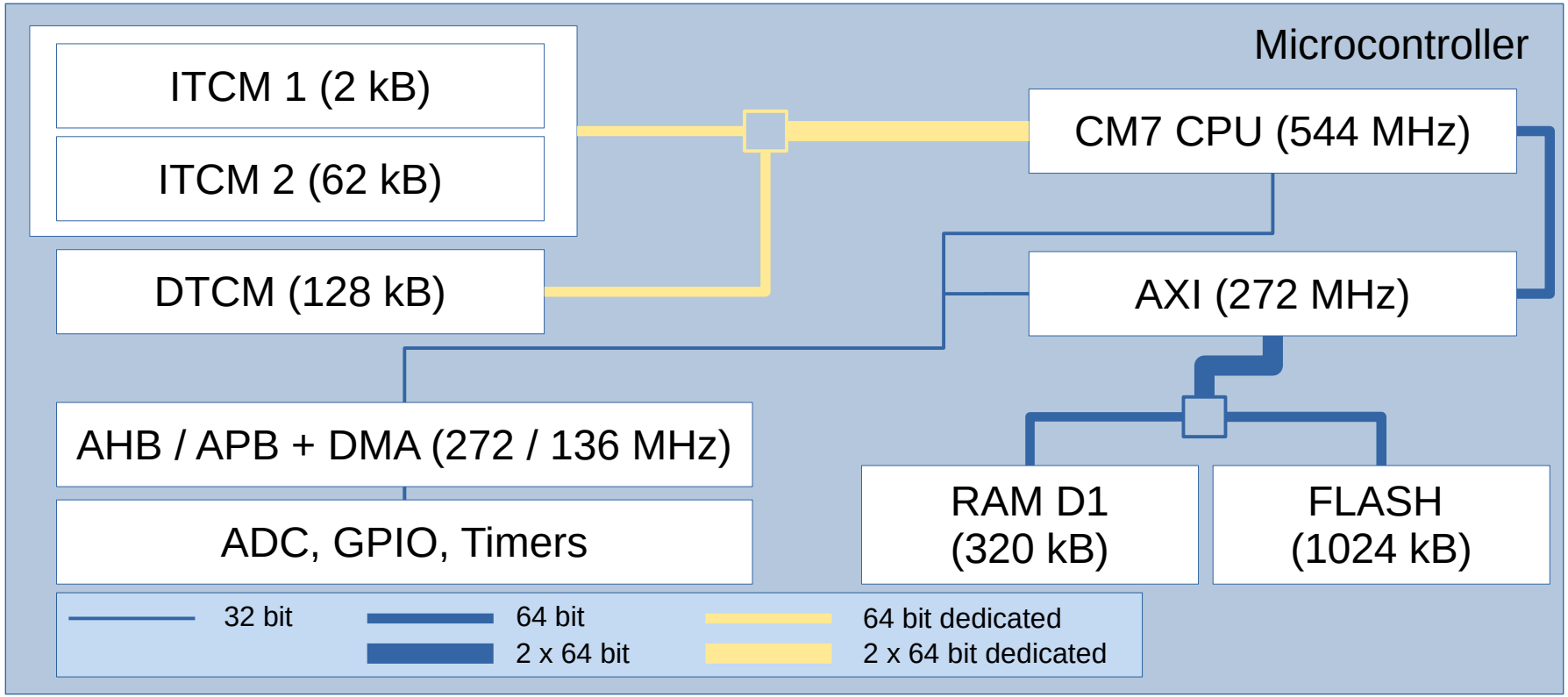
Additional boards (designed in KiCAD):

- *h*=16 cogging module installed near the PS master clock (running since Jan 2022 24/7)
- *h*=1 bucket selector hat installed near an RF measurement front-end (beam tests only)



DP-FPU = double precision floating point unit, ADC = analogue to digital converter, DAC = digital to analogue converter FMC = flexible memory controller, (M)DMA = (master) direct memory access controller, OCTOSPI = 8 x serial parallel interface, FMAC = filter mathematics accelerator, Cordic = coordinate rotation digital computer, PTP = precision timing protocol, ANN = artificial neural network

Parts of the architecture used for synchro



I/DTCM = Instruction/Data Tightly Coupled Memory, CM7 CPU = ARM Cortex M7 Central Processing Unit, AXI = Advanced eXtensible Interface
AH/PB = Advanced High-performance Bus / Peripheral Bus, DMA = Direct Memory Access, USB = Universal Serial Bus, GPIO = General Purpose Input and Output,
RAM D1 = Random Access Memory in the D1 clock domain

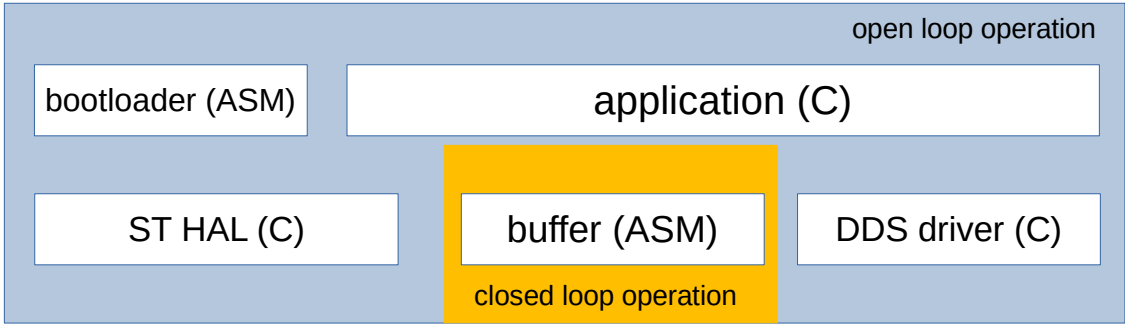
Performance and firmware structure

Operating modes and performance

- closed loop / smooth loop switch
  - externally triggered at 0.5 – 2 MHz from beam control
- open loop / phase measurement
  - clocked at 272 MHz
  - non-linear, phase controlled frequency program: 0.1 ms → 500 ms (adaptive sampling)
  - direct phase measurement for *h*=1 (< 1 degree resolution, no dead zone)

Simple software structure:

- no operating system → NVIC scheduling
- easy to port to other devices



NVIC = ARM's Nested Vector Interrupt Controller, ASM = assembler, DDS = Direct Digital Synthesizer

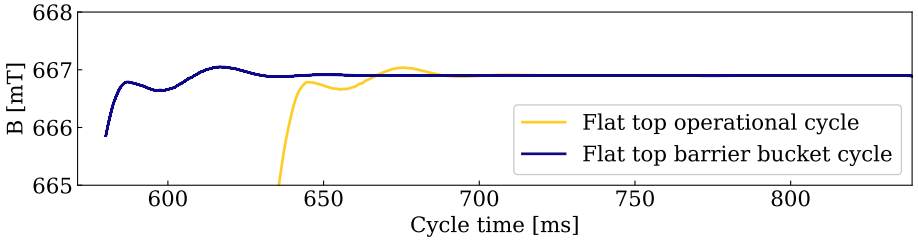
Synchronisation sequence with beam

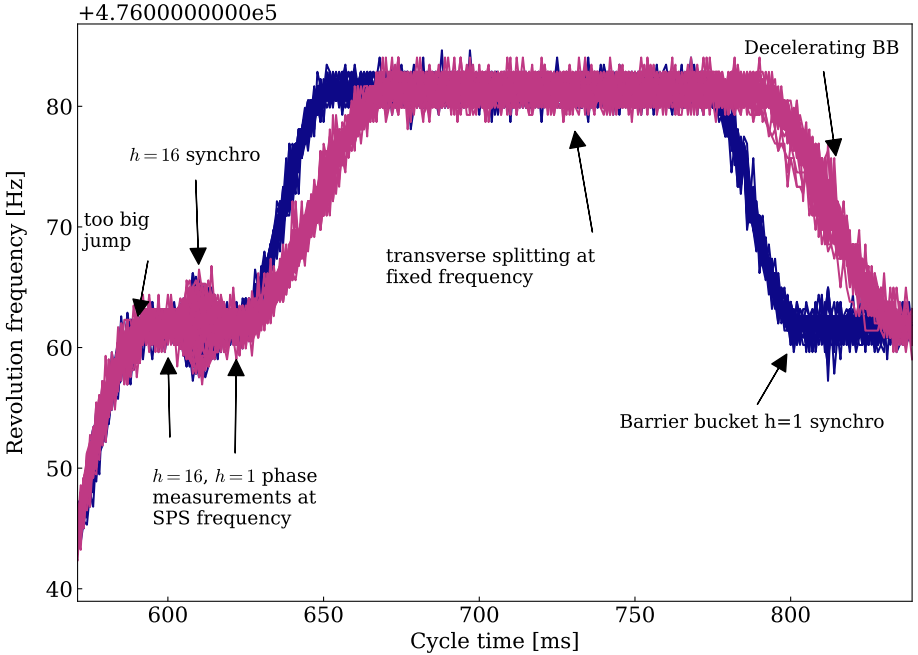
B* and *f_{rev}* plotted from 132 cycles *during setup

- *h*=16 phase measurement
- cogging – different from cycle to cycle – the main synchronisation mechanism
- *h*=1 phase measurement
- fixed frequency for transverse splitting
- back to SPS reference frequency

Caveats

- Frequency jump from closed to open loop
- Barrier bucket handover (static / decelerating)





Conclusions

- Barrier bucket synchronisation scheme devised
- Prototype hardware and firmware made to try the scheme with the present PS beam control
- Barrier bucket synchronisation validated with beam component-by-component:
  - Handover from closed to open loop
  - *h*=16 cogging and *h*=1 bucket selection
  - Radial steering and transverse splitting
  - Radial steering and extraction
- High intensity beam in barrier buckets extracted to SPS and delivered to North Area experiments

Outlook

- Prepare present prototype for operational pilot runs
- Implementation in new beam control, paths to integrate with FPGAs if needed:
  - Controller interfaced with simple FPGA
    → two devices, reuse of implementation, open / closed tools
  - C routines running on a SoC → one device, reuse of C code, closed tools
  - Core C routines implemented in VHDL / Verilog → one device, closed tools

FPGA = field programmable gate array, SoC = system on chip, VHDL = virtual hardware definition language