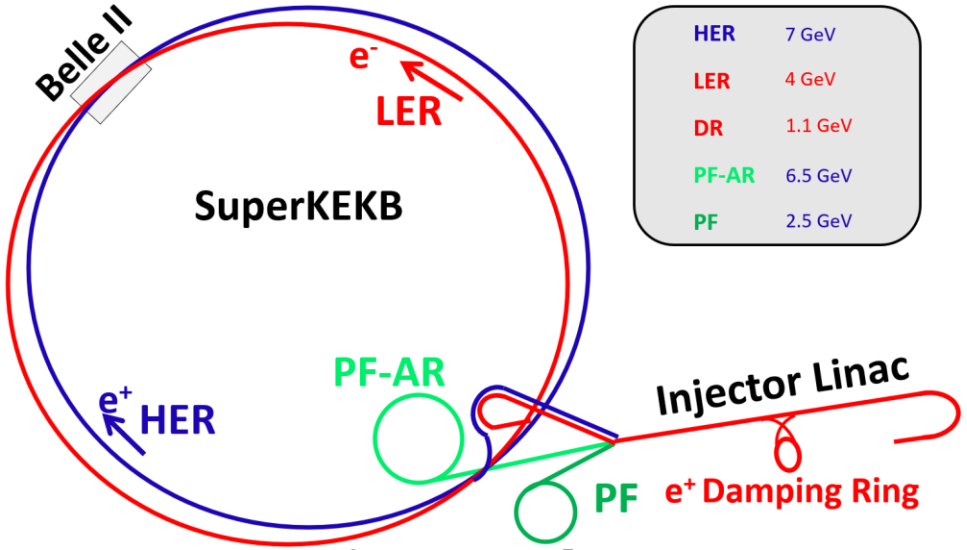
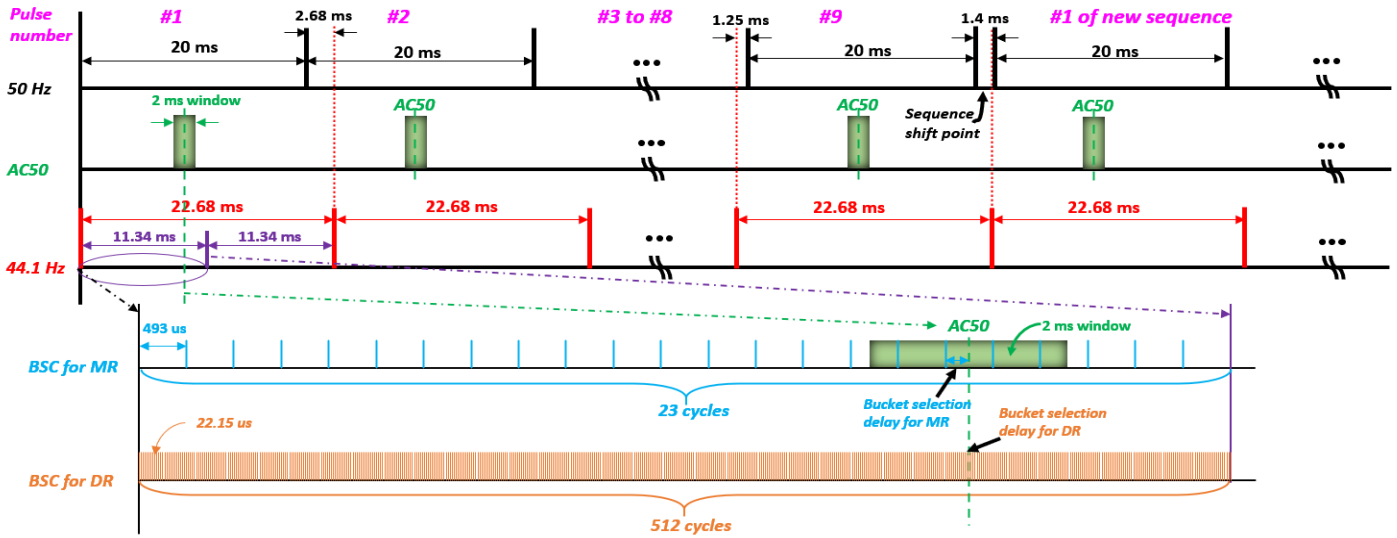
Inter-Generational Compatibility Study of MRF Event Timing Modules

Di Wang (di.wang@kek.jp)
LINAC Control Group
KEK

Current Timing System



Layout of KEK accelerators

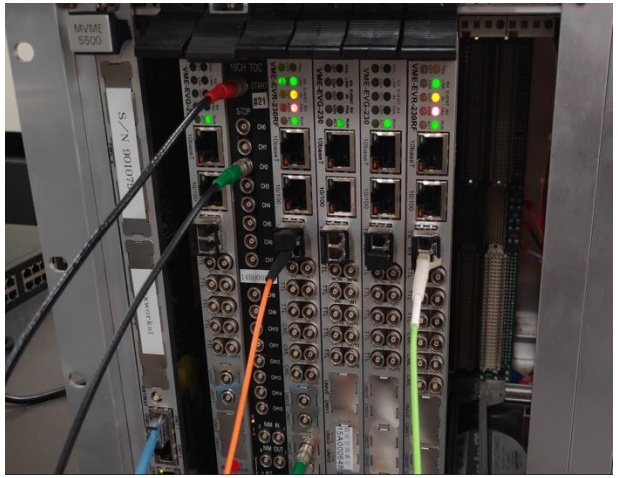


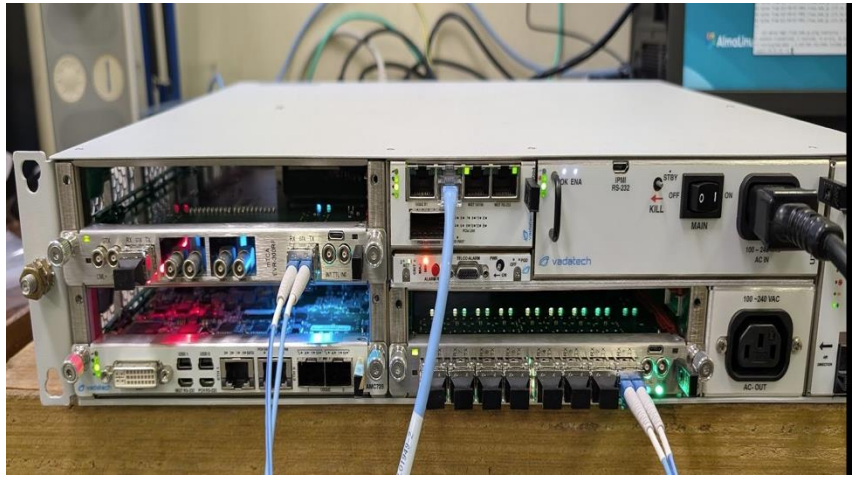
Bucket selection cycle diagram

- The LINAC provides beam simultaneously to four storage rings at 50 Hz
- Timing system must ensure:
  - Stable synchronization with 50 Hz AC line
  - Precise Pulse-to-Pulse Modulation (PPM) every 20 ms
  - Flexible RF bucket selection across injector and multi rings
- The current timing system is highly complex
  - 3 EVGs are cascaded at main station to follow 44.1 Hz BSC and 50 Hz AC line
  - Various EVR types are used (MRF VME/PXI EVR, Virtex6-based embedded EVR, EVO/EVE)

The Timing System Upgrade

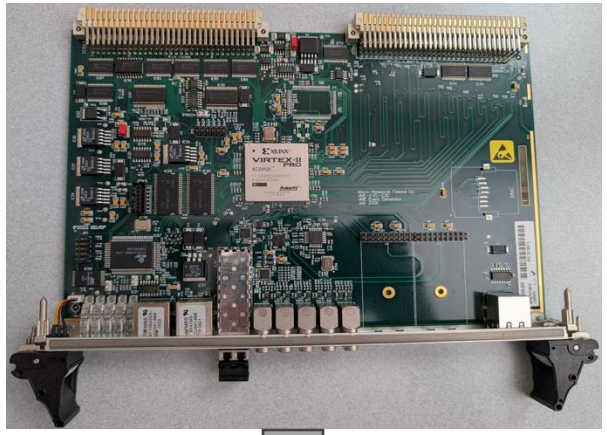
- EVG & EVR from MRF company
- KEK LINAC relies on a large, distributed timing system with over 100 EVRs (mostly VME based) since 2008.
- The VME platform is stable but becoming obsolete; key components like the MVME5500/MVME6100 CPU are being discontinued.
- MRF 230 series EVG & EVR are marked as NLA; migration to 300 series is mandatory
- We decide to migrate our timing system from VME to the modern MicroTCA standard.



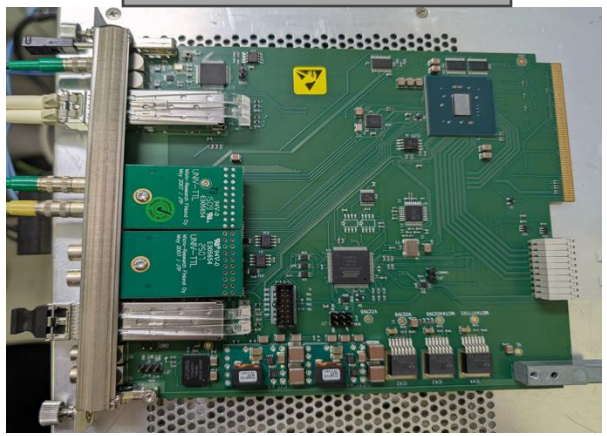


Two Migration Strategy

VME-EVG-230



MTCA-EVR-300



Path A: **Replace EVR first**

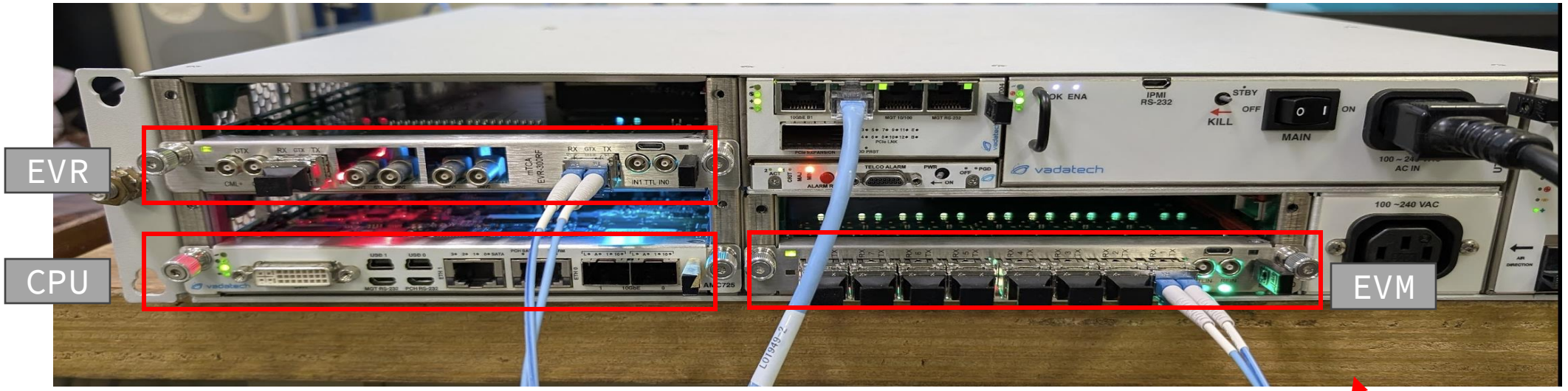
- Goal: fully non-VME timing system
- Chosen approach: **replace EVG first**
  - Remove VME from the “heart” of the timing system
  - Deploy the latest EVG software stack from the start
  - Unlock advanced functionality of the MRF 300 series
  - Simplify subsequent EVR migration and system expansion
- *Verification of compatibility between MTCA-EVM-300 of VME-EVR-230 is critical*

MTCA-EVM-300

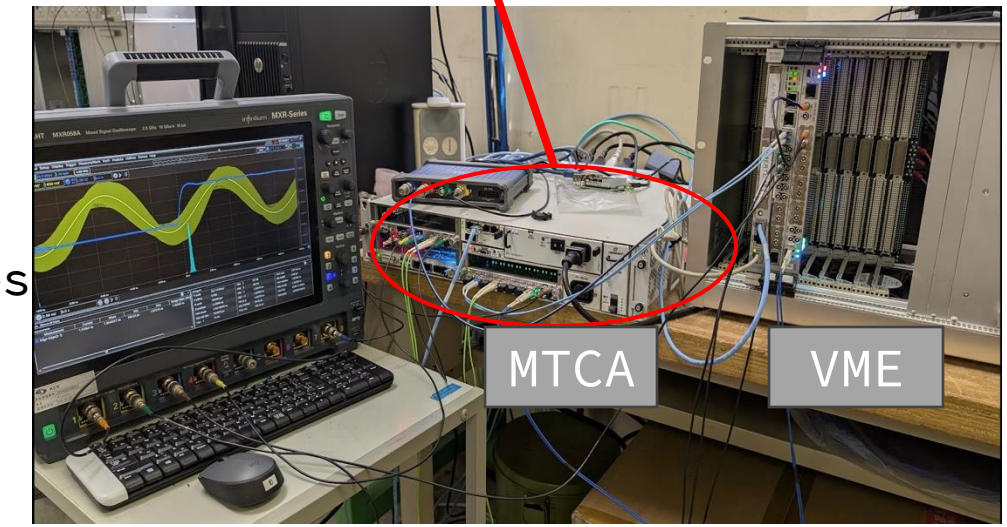
VME-EVR-230

Path B: **Replace EVG first**

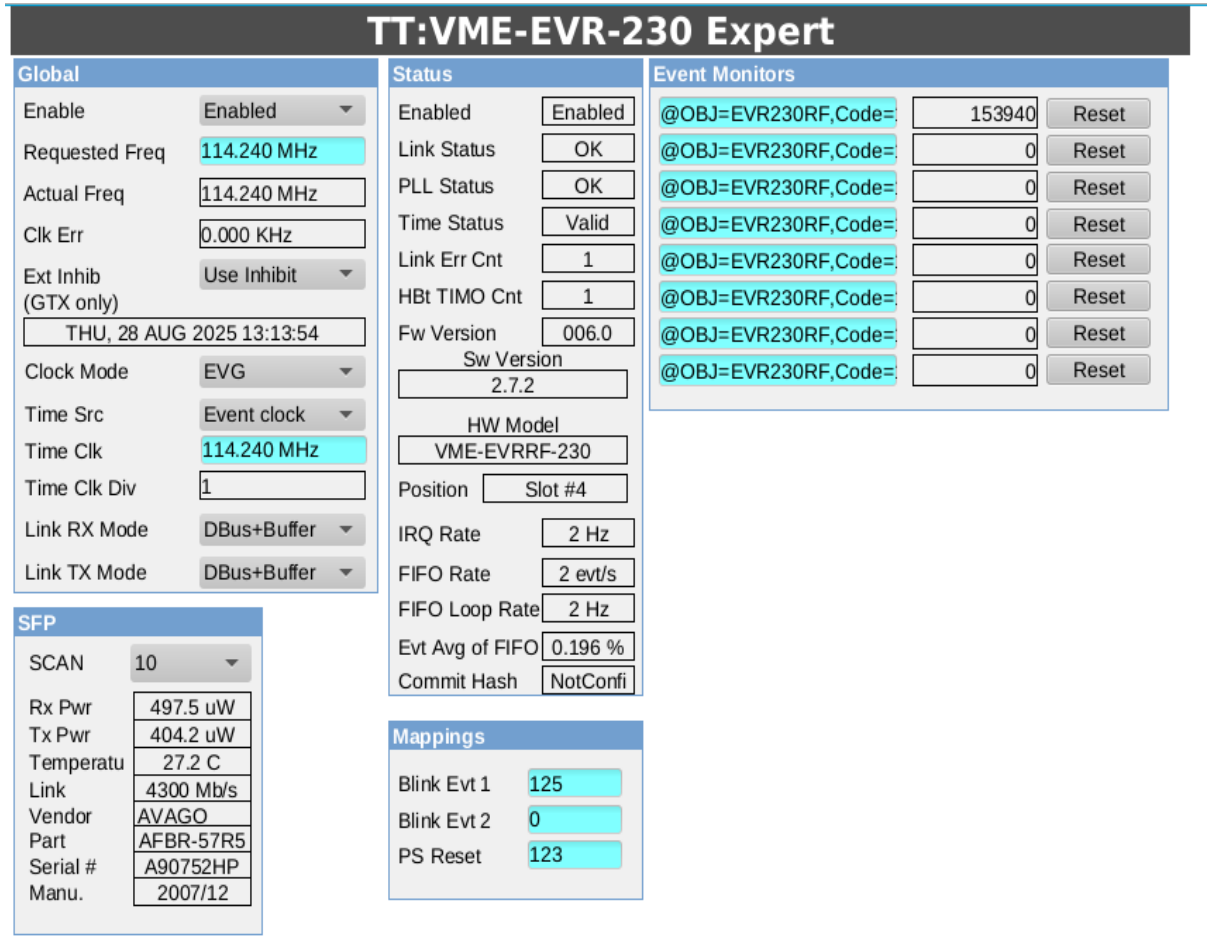
Experimental Setup



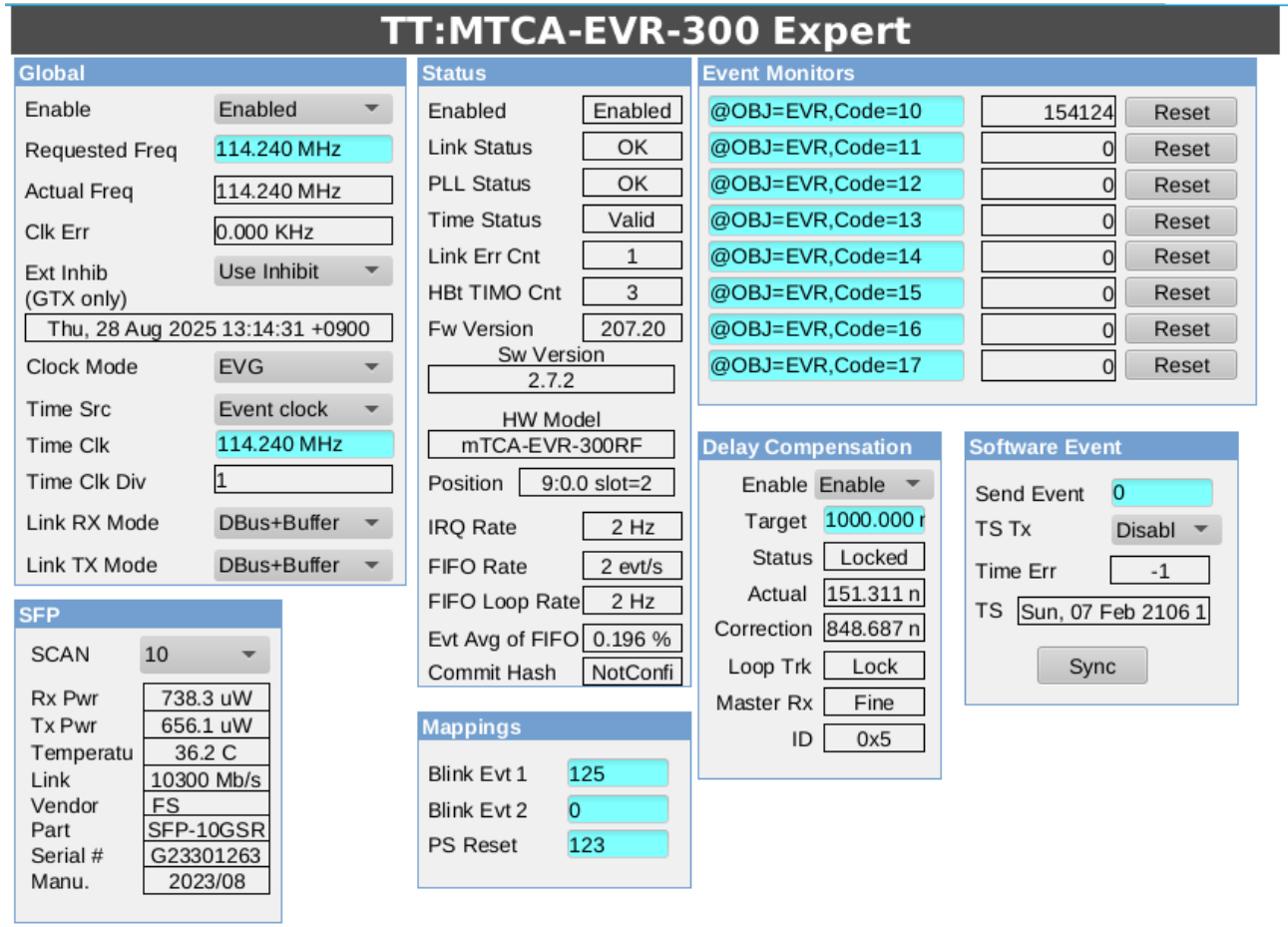
- VadaTech Chassis: VT814
- VadaTech MCH: UTC004
- VadaTech CPU: AMC725
  - AlmaLinux 9
  - EPICS 7 & mrfioc2 (EPICS module) on NFS
  - Single source code -> Linux & VxWroks binaries
- MRF MTCA-EVM-300, MRF MTCA-EVR-300RF, VME-EVR-300, VME-EVR-230RF, PXI-EVR-230
- Keysight Oscilloscope (2.5 GHz, 16 GSa/s)
- Picoscope 4424A & Raspberry Pi 5 (details later)



Control Interface



OPI for VME-EVR-230

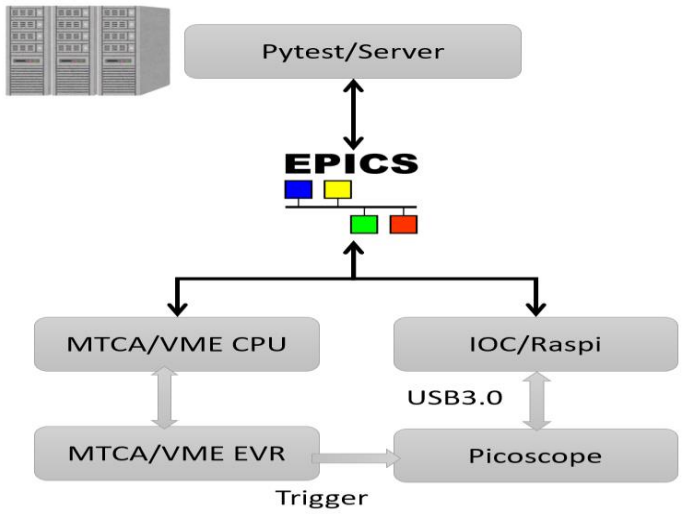


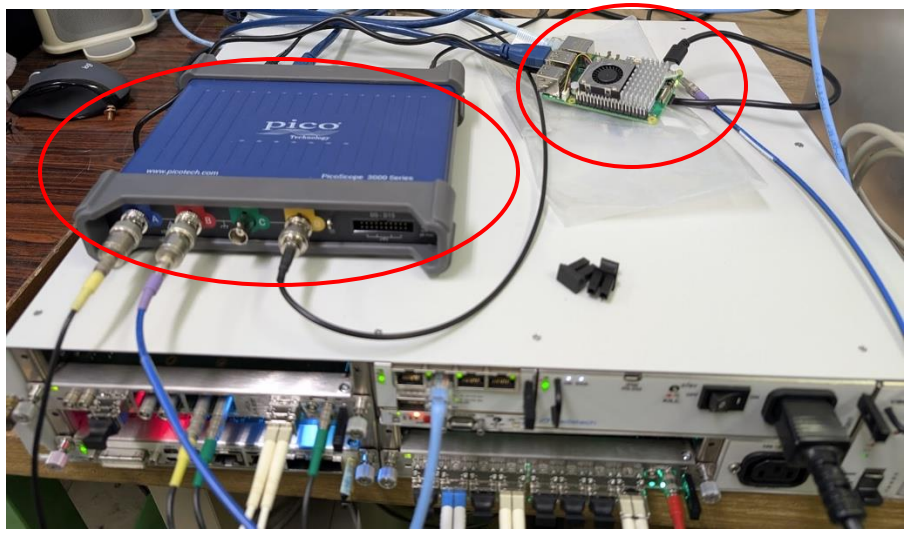
OPI for MTCA-EVR-300

Keep EPICS PV naming and OPI consistency for various of form factors

Basic Function Verification

- Common functions for EVR of different form factors
  - EVR link/TS/SFP status, Pulse-Gen, IO mapping, Output width/polarity
- A pytest-based framework is used to automate the verification
  - Automatically sets up EVR and read status/trigger waveform back
  - Allows flexible configuration through YAML
  - GitLab CI automation and HTML test report generation
  - Works for VME/PXI/MTCA EVRs
- Picoscope connects to Raspberry Pi with USB (both portable)
  - AsynPortDriver based epics driver developed for Picoscope





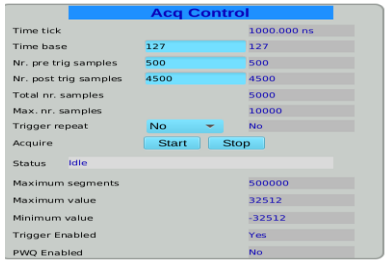
```yaml
mtca-evr-300rf-example:
  prefix: "TT:MTCA-EVR-300:"
  ioc_start: "YES"
  ioc_exe: "/path/to/epics/modules/mrfioc2/bin/linux-x86_64/mrf"
  ioc_cmd: "evr300.cmd"
  cmd_path: "/path/to/epics/ioc/mrf300-test-ioc/mtca"
  hardware: "MTCA-EVR-300RF"
  use_scope: "YES"
  scope_prefix: "TT:PS3405D:"
  scope_port: "FPUV1"
  scope_channel: "A"
  evr_name: "EVR"
  use_pva: "NO"

vme-evr-230-default:
  prefix: "TT:VME-EVR-230:"
  ioc_start: "NO"
  hardware: "VME-EVR-230"
  use_scope: "YES"
  scope_prefix: "TT:PS3405D:"
  scope_port: "FP1"
  scope_channel: "B"
  evr_name: "EVR230"
  use_pva: "NO"
```

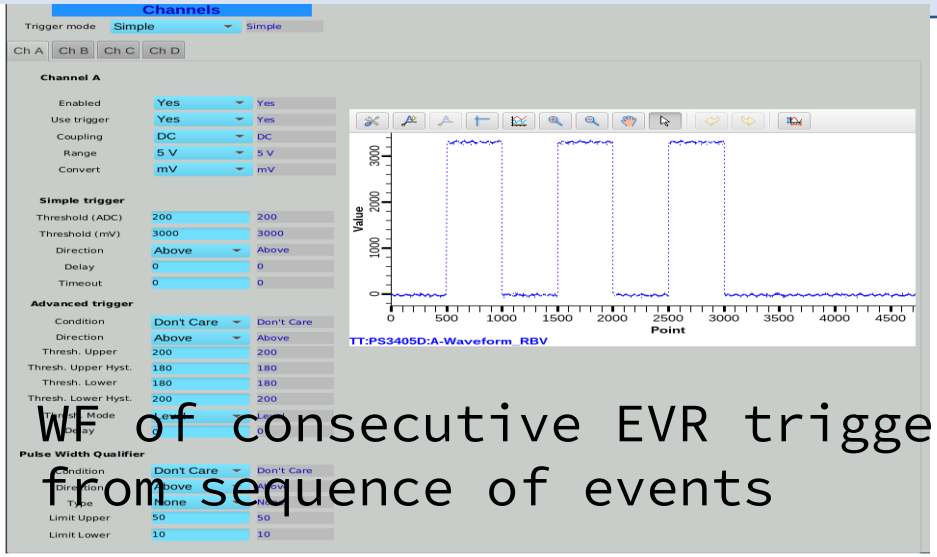
YAML config

Basic Function Verification

```
#!../../bin/linux-arm/testasynps3000a
< envPaths
epicsEnvSet("IOC","ioctestasynps3000a")
epicsEnvSet("TOP","/home/pi/epics/R7.0.9/modules/asynps3000a")
epicsEnvSet("ASYN","/home/pi/epics/R7.0.9/modules/asyn-4-45")
epicsEnvSet("EPICS_BASE","/home/pi/epics/R7.0.9/base")
cd "/home/pi/epics/R7.0.9/modules/asynps3000a"
## Register all support components
dbLoadDatabase "dbd/testasynps3000a.dbd"
testasynps3000a_registerRecordDeviceDriver pdbbase
epicsEnvSet("PP",                          "TT")
epicsEnvSet("DEVICE_NAME",                 "PS3405D")
epicsEnvSet("EPICS_CA_MAX_ARRAY_BYTES",    "10000000")
epicsEnvSet("PREFIX",                      "TT:PS3405D:")
epicsEnvSet("PORT",                        "PICO")
epicsEnvSet("MAX_SAMPLES",                 "10000")
PS3000AConfig("PICO", "10000")
[INF] PS3000A:PS3000A:229 PicoScope 3000A Series Driver
[INF] PS3000A:getInfo:322 device information:
Driver Version: PS3000A Linux Driver, 2.2.153.7541
USB Version: 3.0
Hardware Version: 1
Variant Info: 3405DMSO
Serial: KP361/0179
Cal Date: 08Feb24
Kernel: 0.0
Digital H/W: 1
Analogue H/W: 1
Firmware 1: 1.7.5.0
Firmware 2: 1.0.67.0
dbLoadRecords("db/ps3000a.template", "P=TT:PS3405D:,R=,PORT=PICO,ADDR=0,TIMEOUT=1,MAX_SAMPLES=10000")
dbLoadRecords("db/ps3000aN.template","P=TT:PS3405D:,R=A-,PORT=PICO,ADDR=0,TIMEOUT=1,NAME=Channel A,MAX_SAMPLES=10000")
dbLoadRecords("db/ps3000aN.template","P=TT:PS3405D:,R=B-,PORT=PICO,ADDR=1,TIMEOUT=1,NAME=Channel B,MAX_SAMPLES=10000")
dbLoadRecords("db/ps3000aN.template","P=TT:PS3405D:,R=C-,PORT=PICO,ADDR=2,TIMEOUT=1,NAME=Channel C,MAX_SAMPLES=10000")
dbLoadRecords("db/ps3000aN.template","P=TT:PS3405D:,R=D-,PORT=PICO,ADDR=3,TIMEOUT=1,NAME=Channel D,MAX_SAMPLES=10000")
cd "/home/pi/epics/R7.0.9/modules/asynps3000a/iocBoot/ioctestasynps3000a"
iocInit
Starting iocInit
############################################################################
## EPICS R7.0.9
## Rev. 2025-06-04T18:00+0900
## Rev. Date build date/time:
############################################################################
iocRun: All initialization complete
dbpf "TT:PS3405D:A-Enabled" "1"
```

AsynPortDriver-based IOC on Pi

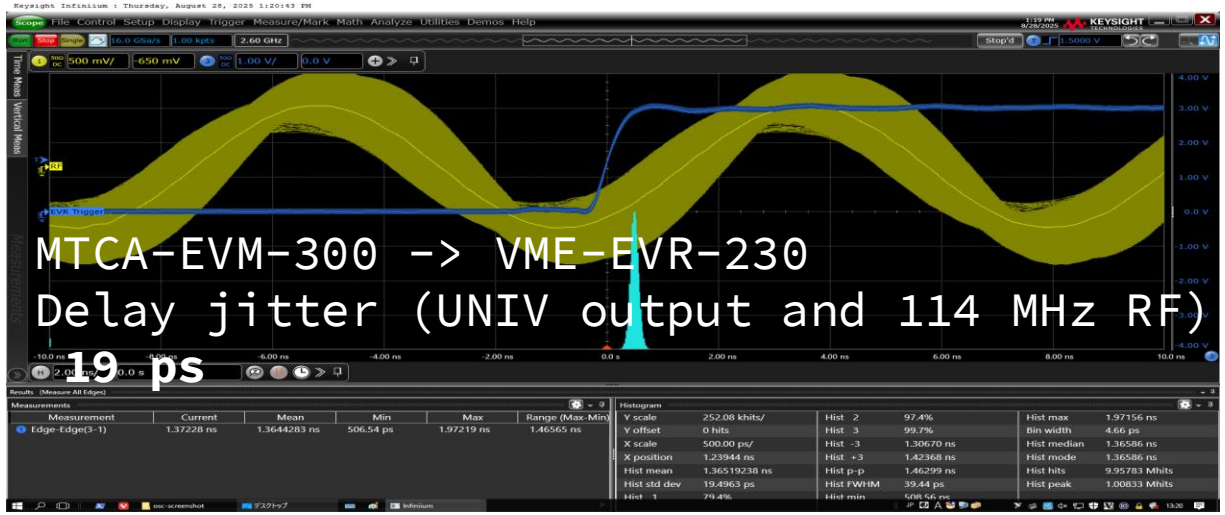


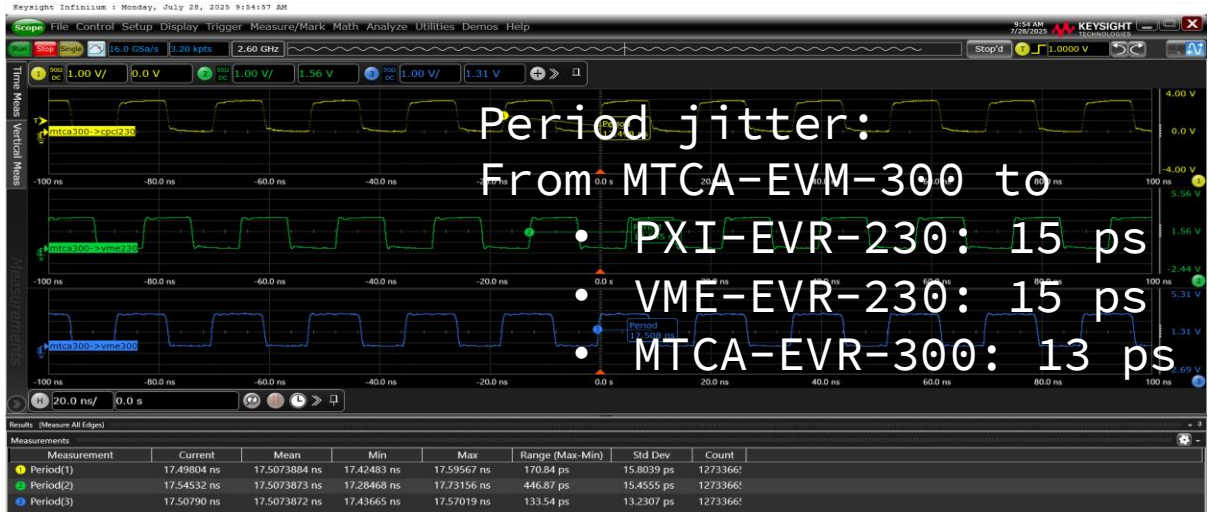


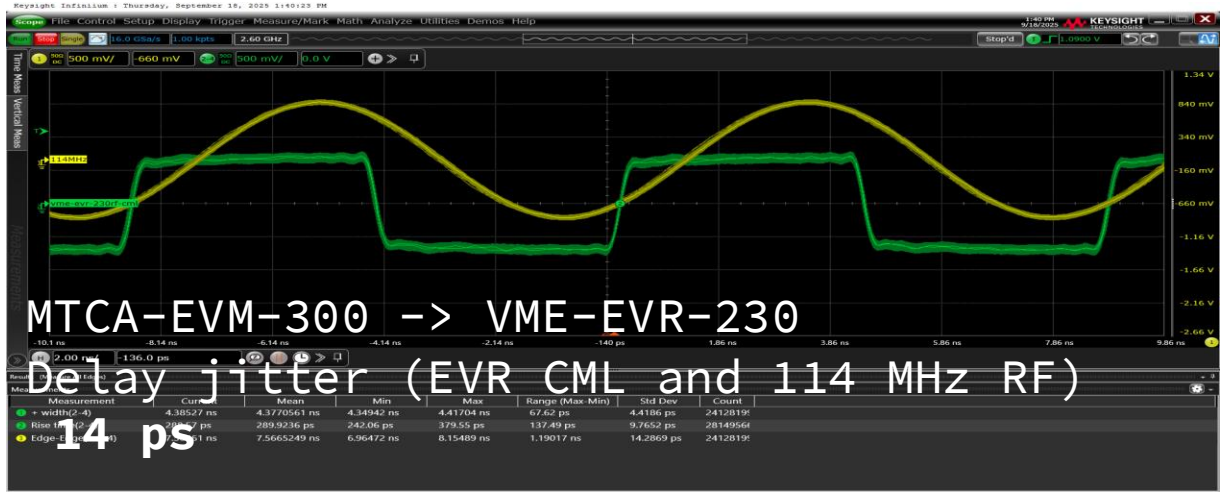
WF of consecutive EVR trigger from sequence of events

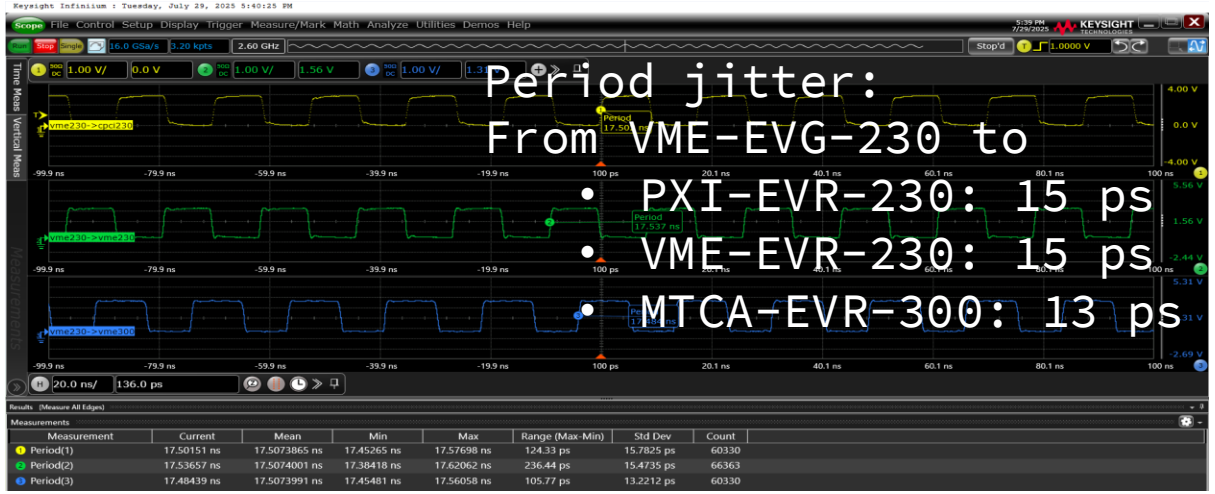
HTML test report generated by GitLab pipeline

Timing Performance









SuperKEKB MR Requirement: <30 ps

Evt Code/Dbuf Long Term Stability

- *Because injection pattern switches frequently, we have occasionally observed missing event codes received by EVRs / data lost from data buffer*
- Python script
  - Simulates operation and calculates injection pattern
    Generates event code/delay waveform
    Applies change to MTCA EVM IOC
  - Daily crontab to verify whether EVR received event code/delay is matching
- VME EVR IOC:
  - Add `EvtLog` iocsh function from mrfioc2, allows changing:
    - NFS directory used to save log files
    - `epicsRingPointer` buffer size
    - format of log file (ascii or binary)
  - Log files hierarchy:
    EVR-NAME/YEAR/MONTH/DAY/HOUR.log
  - Log data format:
    `code-second.nanosecond`
- Similarly, 50 Hz data buffer is also logged and verified
- *No code mismatch or timing violation are observed*

```
=== Start Event Sequence Checker ===
Start processing EVR230RF/2025/09/15/00.log
Initial Phase: Found event 38: 38-1757861999.560603781
Start processing EVR230RF/2025/09/15/01.log
Start processing EVR230RF/2025/09/15/02.log
Start processing EVR230RF/2025/09/15/03.log
Start processing EVR230RF/2025/09/15/04.log
Start processing EVR230RF/2025/09/15/05.log
Start processing EVR230RF/2025/09/15/06.log
Start processing EVR230RF/2025/09/15/07.log
Start processing EVR230RF/2025/09/15/08.log
Start processing EVR230RF/2025/09/15/09.log
Start processing EVR230RF/2025/09/15/10.log
Start processing EVR230RF/2025/09/15/11.log
Start processing EVR230RF/2025/09/15/12.log
Start processing EVR230RF/2025/09/15/13.log
Start processing EVR230RF/2025/09/15/14.log
Start processing EVR230RF/2025/09/15/15.log
Start processing EVR230RF/2025/09/15/16.log
Start processing EVR230RF/2025/09/15/17.log
Start processing EVR230RF/2025/09/15/18.log
Start processing EVR230RF/2025/09/15/19.log
Start processing EVR230RF/2025/09/15/20.log
Start processing EVR230RF/2025/09/15/21.log
Start processing EVR230RF/2025/09/15/22.log
Start processing EVR230RF/2025/09/15/23.log
=== Event Sequence Checker Summary ===
Tick frequency: 114.24000 MHz (period ~8.753501 ns)
Sequence length: 132 events
Sequence period: 223.885373 ms
Files scanned: 24
Total events processed: 47518944
Code mismatches: 0
Timing violations: 0 (tolerance 10.0 ns)
Resyncs applied: 0
```

Daily event code checking

Summary & Future Work

- Upgrade plan
  - Goal is a fully non-VME timing system with MTCA & MRF 300 modules
  - Phased migration approach ensures safety and compatibility
- Compatibility validated
  - Event code distribution tested and basic functions confirmed
  - Jitter performance within requirements – no degradation observed
  - Long term event code/data buffer transmission stability verified
- Next migration tasks
  - Replace other VME-based timing modules (TDC/RFM) with MTCA ones
  - New Linux driver for MTCA TDC/RFM modules
  - Integrate into EPICS and conduct system-level tests