TEST AUTOMATION UTILITY FOR EVENT TIMING MODULES

D. Wang[1]*, M. Satoh[1], [1] KEK, Ibaraki 305-0801, Japan

Abstract

The timing module plays a crucial role by providing precise triggering signals to various accelerator devices, controlling their operational states under different beam modes, and offering synchronized timestamps. The long-term operational stability of these systems requires a comprehensive stability testing of the timing module during the development phase. However, updates of FPGA firmware, operating systems, kernel drivers, or epics driver can potentially interfere the performance and stability of the timing module. To address this challenge, this study developed an automated testing utility capable of quickly and efficiently verifying key functionalities of the timing module. These functionalities include the reception of event codes, consistency of delay and width setting of front panel output pulses, and so on. This automated utility not only enhances development efficiency but also strengthens reliability verification throughout the software iteration process, significantly improving overall system performance and stability. Through this research, we demonstrate the application value and practical effectiveness of automated testing in the development of accelerator system timing modules.

INTRODUCTION

The precise synchronization of various accelerator devices is critical for the reliable and stable operation of particle accelerators. This is achieved through a distributed timing system, which provides synchronized event codes and trigger pulses to control devices such as magnets and beam position monitors. At the heart of this system are event timing modules, which must operate with high precision and stability. The long-term performance of these modules is a core requirement for accelerator operation. However, the continuous evolution of hardware and software—including updates to FPGA firmware, operating systems, kernel drivers, or EPICS drivers can introduce changes that affect a module's performance and stability.

The KEK LINAC is currently undergoing a significant timing system upgrade, migrating from an aging VME-based platform to the modern MicroTCA (MTCA) standard. This transition involves a complex, mixed-generation environment where new and legacy modules must coexist. Manually validating hundreds of these timing modules in a dynamic environment is a time-consuming and error-prone process that lacks the necessary repeatability.

To address these challenges, we have developed a robust, automated Site Acceptance Test (SAT) utility. This system is designed to quickly and efficiently verify the key functionalities of timing modules. This paper details the architecture and implementation of this automated utility, demonstrating how it enhances development efficiency, improves reliability, and ensures the operational integrity of the timing system during this crucial migration.

[*] di.wang@kek.jp

THE TIMING SYSTEM UPGRADE

Timing System at KEK LINAC

The LINAC's current timing infrastructure is built upon a large, distributed system comprising more than 100 event receiver (EVR) modules, mainly based on the VME standard. However, the VME platform is facing obsolescence problem. Critical components, such as the MVME5500 and MVME6100 CPU modules, are no longer being manufactured. To address this, we are undertaking a comprehensive migration of our timing system from the VME standard to the modern MicroTCA (MTCA) platform. This significant change requires a rigorous and thorough validation process for all new hardware components.

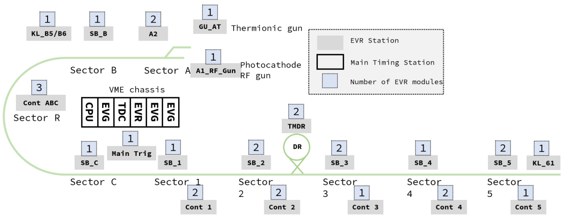


Figure 1: The timing station and number of event receivers at KEK LINAC.

Figure 1 shows 21 distributed timing stations all over LINAC. The central and most critical of these is the main trigger station, which contains three VME event generators (EVGs) and one VME event receiver (EVR) [1]. The number of VME EVRs deployed at each station is also indicated in the figure. Beyond this, the system incorporates different types of EVRs to support various subsystems. For example, the pulsed magnet control system is equipped with 18 PXI EVRs [2]. The total count of EVRs across all stations and subsystems exceeds 100.

Event Timing Modules

The event timing system utilizes a range of hardware platforms from Micro-Research Finland (MRF), including VME, PXI, PCIe, and MTCA form factors [3]. Figure 2 shows the MicroTCA EVR 300 module. This modular architecture accommodates the diverse needs of the accelerator subsystems.

The timing modules are a continuously evolving technology. Newer module series, such as the 300 series, offer significant enhancements over earlier versions. These improvements include active delay compensation, segmented data buffers, and flip-flop outputs, enabling more flexible and precise pulse generation. Continuous updates to firmware

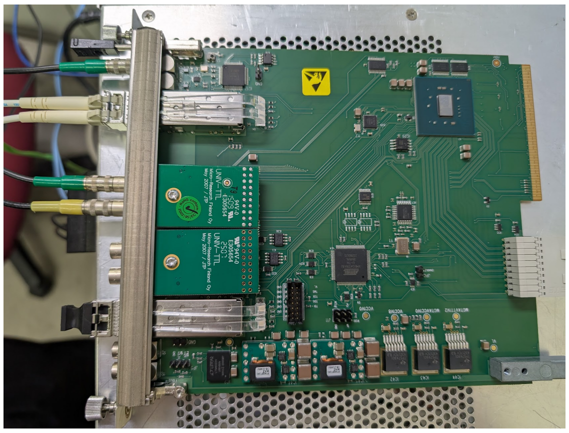


Figure 2: The MicroTCA EVR 300 module.

and EPICS drivers further improve performance and fix bugs within each module series. Every new software release may introduce functionality that requires revalidation to ensure system stability and reliability.

Table 1: Possible Combination of Event Module Connections

Event generator	Event receiver
VME-EVG-230	VME-EVR-230
VME-EVG-230	VME-EVR-230RF
VME-EVG-230	PXI-EVR-230
VME-EVG-230	MTCA-EVR-300RF
VME-EVG-230	VME-EVR-300
VME-EVG-230	Home-made EVR
MTCA-EVM-300	VME-EVR-230
MTCA-EVM-300	VME-EVR-230RF
MTCA-EVM-300	PXI-EVR-230
MTCA-EVM-300	MTCA-EVR-300RF
MTCA-EVM-300	VME-EVR-300
MTCA-EVM-300	Home-made EVR

This ongoing evolution creates a need for rigorous compatibility testing. A key aspect of our system's migration is the validation of various EVG and EVR combinations. Table 1 shows several possible combinations. For example, our current configuration utilizes VME-EVG-230 series modules in conjunction with 230 series VME and PXI EVRs. The future transition to the MTCA standard may involve connecting a newer MTCA-EVM-300 series to the existing VME-EVR-230 series modules. Successful integration of these hardware generations requires extensive testing to confirm their functional compatibility.

Challenges

The transition of the timing system involves phasing out obsolete VME hardware, which creates a complex, mixed-generation environment. A key bottleneck is the manual validation of hundreds of timing modules, a process that is time consuming, prone to human error, and lacks repeatability.

To address these challenges, we are developing a robust, automated Site Acceptance Test suite. This solution must:

- Increase test coverage and consistency.
- Reduce manual intervention and testing time.
- Generate automated test reports for validation and records.

This automated solution is crucial for ensuring the reliable and timely migration of our timing system to the modern MTCA standard while maintaining the operational integrity of the accelerator.

TEST AUTOMATION UTILITY

Test Automation Architecture

Figure 3 shows the architecture of the test automation utility. It is structured around a Python-based software architecture and modern hardware platforms. A Python-based test script, executed on a Linux server, controls the test flow. This script interfaces with the EVRs and an oscilloscope via EPICS, using the oscilloscope to precisely measure the EVR's output signal.

The hardware of the test system consists of VadaTech MTCA products [4]. An EPICS Input/Output Controller (IOC) running on a Raspberry Pi serves as the control computer for oscilloscope.

For software management, uv is utilized to handle Python versions and package dependencies, ensuring a reproducible environment [5]. The pytest framework provides a robust and scalable foundation for the entire test automation suite.

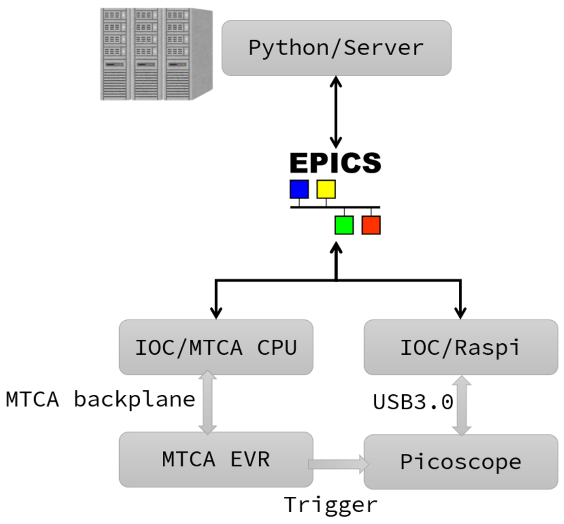


Figure 3: Architecture of test automation utility.

Framework Design

The framework design is based on an object-oriented approach using three key Python classes. The IOCBase class serves as a foundational class for inheritance, providing methods for interacting with EPICS Process Variables (PVs).

The other two classes, EVRDevice and ScopeDevice, are specialized classes that inherit from IOCBase to manage the specific functionalities of the EVR and the oscilloscope, respectively.

The tests are organized into three distinct Python files to ensure a logical and modular structure. The test_basic.py contains tests for fundamental functions and EVR status checks. The test_fpout.py is dedicated to validating the generation of trigger outputs from the front panel. Finally, the test_fpinp.py is used to validate the generation of interrupts from front panel input signals. This organization allows for clear separation of test responsibilities and simplifies test maintenance.

Configurable Tests

To further increase flexibility, many test setups are now configurable. A human-readable YAML file is used for these configurations, allowing easy adjustments without modifying the test script itself [6].

The provided example shows some of the configurable items:

- PV Prefixes: These define the unique EPICS PV prefixes for the EVR and the oscilloscope IOC.
- IOC Management: User can choose whether to start a new IOC using a Python subprocess or connecting to an existing one.
- IOC Startup: If user choose to start a new IOC, user can specify the IOC binary path, startup command, and startup directory.
- Hardware Connections: The configuration maps which EVR output ports are connected to which oscilloscope channels (e.g., front panel universal output 1 of EVR connects to oscilloscope channel A).
- EPICS Protocol: User can specify whether to use PV Access or Channel Access to connect to the IOC.

Basic EVR Function Tests

The basic EVR function tests are designed to verify the fundamental operational capabilities of the module. Example test items include:

- Event Code Receiving: This test validates the EVR's ability to receive and process specific event codes. It involves configuring an internal event code counter to trigger on a designated event and then verifying that the counted rate matches the expected value.
- Heartbeat and Status: The system's health is monitored by polling key status PVs. This ensures that the module is responsive and reporting its operational status without any errors, confirming its basic functionality and connectivity.

IO Validation Tests

This test confirms that a specific event code received by the EVR produces the correct pulse output at its physical output ports. This requires careful synchronization between the event generation and waveform capture. An oscilloscope is armed to capture the waveform of EVR output. The waveform characteristics (e.g., pulse width and output polarity) should match the configured parameters.

IO Tests Using Picoscope

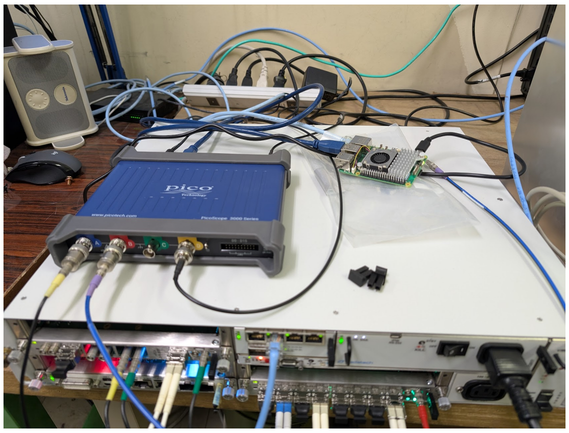
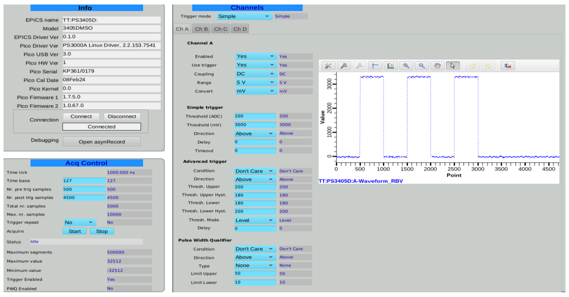


Figure 4: The hardware connection of IO Tests

To capture the waveform using EPICS, Picoscope 3405D is selected as monitoring device because of its Linux support and simple USB connection. As Fig. 4 shows, the Picoscope connects to Raspberry Pi 5 where EPICS IOC runs on. An asynPortDriver based EPICS module is developed to integrate into EPICS environment. To simplify the usability, a OPI panel is created to setup parameters of Picoscope as well as verify the captured waveform. Figure 5 shows the OPI for Picoscope.



Info

EPICS name	TT:PS3405D:
Model	3405DMSO
EPICS Driver Ver	0.1.0
Pico Driver Ver	PS3000A Linux Driver, 2.2.153.7541
Pico USB Ver	3.0
Pico HW Ver	1
Pico Serial	KP361/0179
Pico Cal Date	08Feb24
Pico Kernel	0.0
Pico Firmware 1	1.7.5.0
Pico Firmware 2	1.0.67.0
Connection	Connect Disconnect Connected
Debugging	Open asynRecord

Acq Control

Time tick		1000.000 ns
Time base	127	127
Nr. pre trig samples	500	500
Nr. post trig samples	4500	4500
Total nr. samples		5000
Max. nr. samples		10000
Trigger repeat	No	No
Acquire	Start Stop	
Status	Idle	
Maximum segments		500000
Maximum value		32512
Minimum value		-32512
Trigger Enabled		Yes
PWQ Enabled		No

Channels

Trigger mode: Simple / Simple

Ch A | Ch B | Ch C | Ch D

Channel A		
Enabled	Yes	Yes
Use trigger	Yes	Yes
Coupling	DC	DC
Range	5 V	5 V
Convert	mV	mV
Simple trigger		
Threshold (ADC)	200	200
Threshold (mV)	3000	3000
Direction	Above	Above
Delay	0	0
Timeout	0	0
Advanced trigger		
Condition	Don't Care	Don't Care
Direction	Above	Above
Thresh. Upper	200	200
Thresh. Upper Hyst.	180	180
Thresh. Lower	180	180
Thresh. Lower Hyst.	200	200
Thresh. Mode	Level	Level
Delay	0	0
Pulse Width Qualifier		
Condition	Don't Care	Don't Care
Direction	Above	Above
Type	None	None
Limit Upper	50	50
Limit Lower	10	10

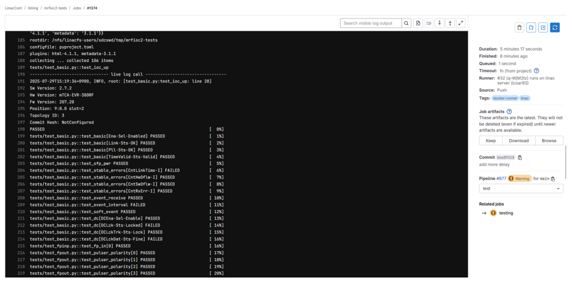
Figure 5: The GUI panel shows Picoscope captures the output pulse from an EVR.

GitLab Automation

To further improve the automation testing process, this suite is integrated with GitLab environment. The hardware installation and cable connections are performed as a prerequisite for testing. Then the entire test workflow is automated through a GitLab CI/CD pipeline, which is triggered upon committing changes to a GitLab repository, such as adding new tests or modifying the configuration.

The automation process is initiated by specifying the IOC name using GitLab CI/CD variables. A GitLab Runner then picks up the assigned task and prepares the test environment. This preparation includes logging into the MTCA server where the hardware resides. An optional step is to start the IOC via a Python subprocess if it is not already running.

The runner then executes the tests as defined in the pipeline. Upon completion, it collects the test artifacts, such as the automatically generated reports, and uploads them back to GitLab , providing a streamlined and auditable workflow from code commit to result analysis.


LinacCont / timing / mrfioc2-tests / Jobs / #1374
Search visible log output
Keep
Download
Browse

```
'4.1.1', 'metadata': '3.1.1'}}
rootdir: /nfs/linacfs-users/sdcswd/tmp/mrfioc2-tests
configfile: pyproject.toml
plugins: html-4.1.1, metadata-3.1.1
collecting ... collected 106 items
tests/test_basic.py::test_ioc_up
------------------------------ live log call -------------------------------
2025-07-29T15:19:36+0900, INFO, root: [test_basic.py:test_ioc_up: line 28]
Sw Version: 2.7.2
Hw Version: mTCA-EVR-300RF
Fw Version: 207.20
Position: 9:0.0 slot=2
Topology ID: 3
Commit Hash: NotConfigured
PASSED                                                                    [  0%]
tests/test_basic.py::test_basic[Ena-Sel-Enabled] PASSED                   [  1%]
tests/test_basic.py::test_basic[Link-Sts-OK] PASSED                       [  2%]
tests/test_basic.py::test_basic[Pll-Sts-OK] PASSED                        [  3%]
tests/test_basic.py::test_basic[TimeValid-Sts-Valid] PASSED               [  4%]
tests/test_basic.py::test_sfp_pwr PASSED                                  [  5%]
tests/test_basic.py::test_stable_errors[CntLinkTimo-I] FAILED             [  6%]
tests/test_basic.py::test_stable_errors[CntHwOflw-I] PASSED               [  7%]
tests/test_basic.py::test_stable_errors[CntSwOflw-I] PASSED               [  8%]
tests/test_basic.py::test_stable_errors[CntRxErr-I] PASSED                [  9%]
tests/test_basic.py::test_event_receive PASSED                            [ 10%]
tests/test_basic.py::test_event_interval FAILED                           [ 11%]
tests/test_basic.py::test_soft_event PASSED                               [ 12%]
tests/test_basic.py::test_dc[DCEna-Sel-Enable] PASSED                     [ 13%]
tests/test_basic.py::test_dc[DCLck-Sts-Locked] FAILED                     [ 14%]
tests/test_basic.py::test_dc[DCLckTrk-Sts-Lock] PASSED                    [ 15%]
tests/test_basic.py::test_dc[DCLckDat-Sts-Fine] FAILED                    [ 16%]
tests/test_fpinp.py::test_fp_in[0] PASSED                                 [ 16%]
tests/test_fpout.py::test_pulser_polarity[0] PASSED                       [ 17%]
tests/test_fpout.py::test_pulser_polarity[1] PASSED                       [ 18%]
tests/test_fpout.py::test_pulser_polarity[2] PASSED                       [ 19%]
tests/test_fpout.py::test_pulser_polarity[3] PASSED                       [ 20%]
```

Duration: 5 minutes 17 seconds
Finished: 8 minutes ago
Queued: 1 second
Timeout: 1h (from project)
Runner: #32 (q-WjM2tz) runs on linac server (lcsar93)
Source: Push
Tags: docker-runner linac

Job artifacts
These artifacts are the latest. They will not be deleted (even if expired) until newer artifacts are available.

Commit bbe89318
add more delay

Pipeline #877 Warning for main
test

Related jobs
→ testing

Figure 6: The test scripts runs on GitLab pipeline.

CONCLUSION

Our automated Site Acceptance Test suite has yielded significant benefits, addressing the key challenges of the manual testing process. The execution time for a full SAT has been reduced from several hours of manual work to just a few minutes, drastically accelerating the validation process. By eliminating the potential for human error, the system ensures that every new module is tested against the exact same criteria, guaranteeing consistent and trustworthy results. The suite also automatically generates detailed HTML reports, which provide a permanent, time-stamped record of test results for every device, essential for quality assurance and long-term maintenance records. The framework is designed for flexibility, making it easily extensible to support timing modules in other form factors beyond MicroTCA.

REFERENCES

[1] D. Wang *et al.*, "Analysis and stabilization of AC line synchronized timing system for superKEKB", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 1015, p. 165 766, 2021. doi:10.1016/j.nima.2021.165766

[2] M. Satoh, D. Wang, and I. Satake, "KEK e-/e+ INJECTOR LINAC CONTROL SYSTEM", in *Proceedings of the 21st Annual Meeting of Particle Accelerator Society of Japan*, pp. 946–950, 2024. https://www.pasj.jp/web_publish/pasj2024/proceedings/PDF/FRP0/FRP039.pdf

[3] "MRF". (2025), http://www.mrf.fi/

[4] "vadatech". (2025), https://www.vadatech.com/

[5] "uv". (2025), https://docs.astral.sh/uv/

[6] "YAML". (2025), https://yaml.org/