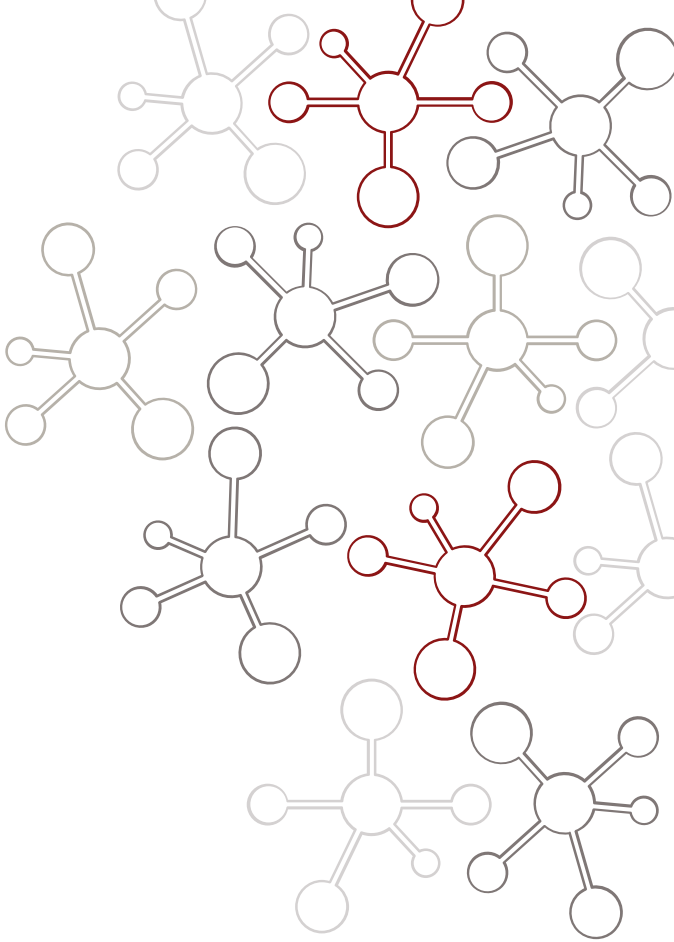
Timing Pattern Generator (TPGGUI) Development

Status Update 1

Alexander Ng / Software Developer / EED/SWE-OPS

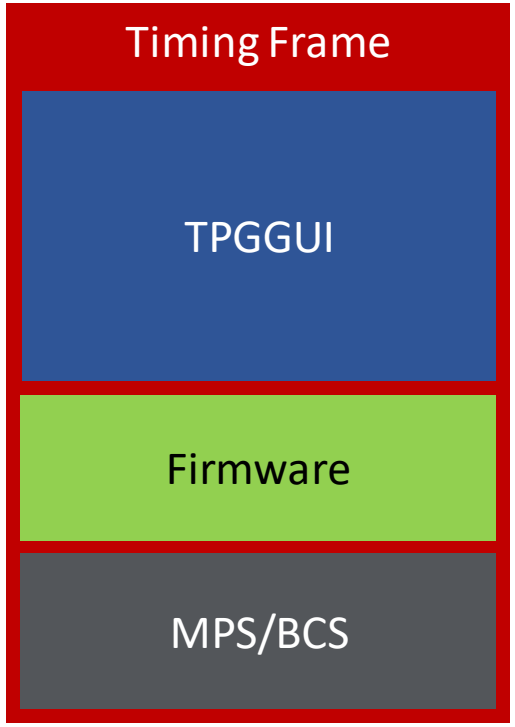
June 15, 2023



What is a TPGGUI?

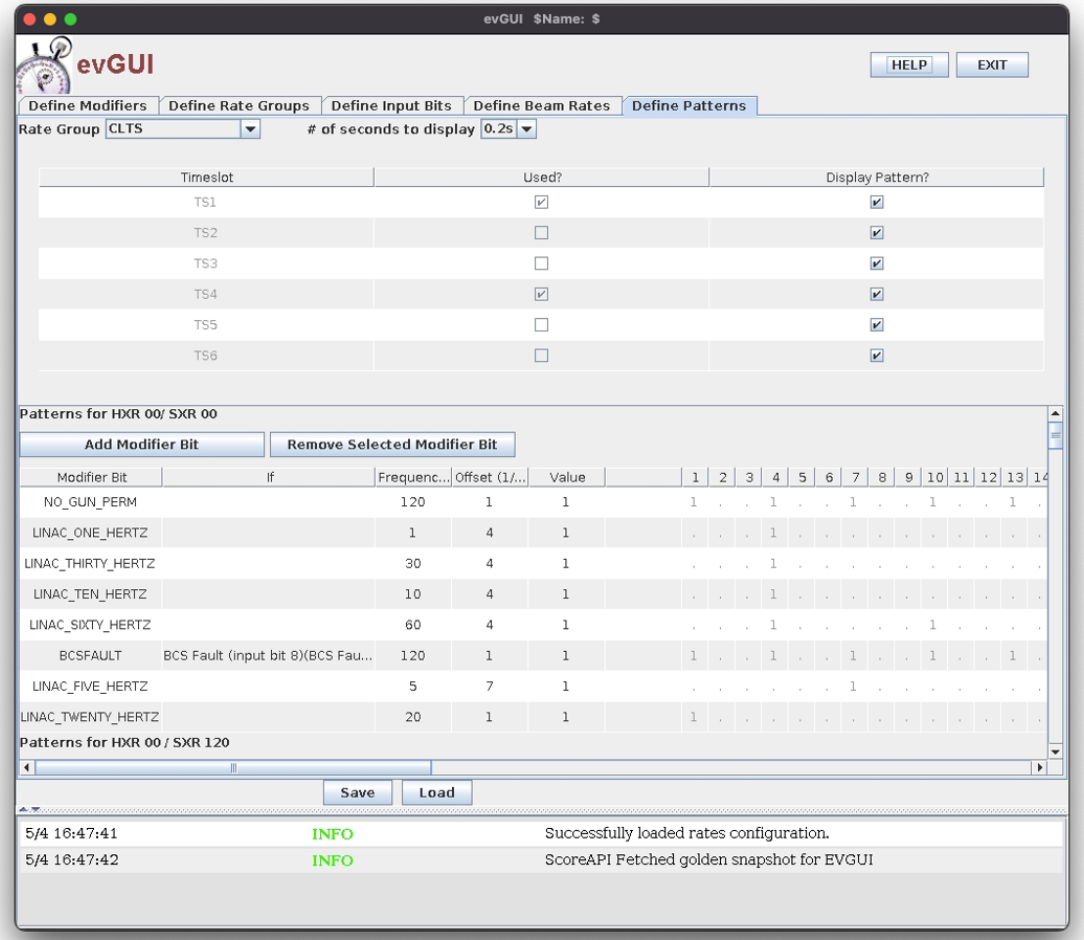
Timing Pattern Generator (TPG) Graphical User Interface (GUI)

- The timing system sends out a timing pattern that contains relevant information to trigger the LASER system to generate beam and to the timing clients and devices along the SC LINAC.
- The timing pattern consists of a sequence of timing frames, which contains data set by firmware and software.
- The firmware defines information that is constantly sent out to identify frames, such as rate specific markers, pulse ID. Other bits in the timing frame are machine operation sensitive and need to be configured at a higher level.
- The TPGGUI will be the software tool used to program timing frames, updating information to the timing pattern to fulfill Operation requirements.
- The TPGGUI will allow operations to program more complex patterns and schedule beam going to multiple destinations while meeting machine and personnel safety requirements.



Motivation for TPGGUI

- Initially, the timing patterns were defined by python scripts in a GitHub repo. This pre-existing repo only handled single destination patterns without a visual/user friendly interface to manipulate the data or view possible conflicts on beam pulses assignments.
- The Copper (Cu/NC) LINAC timing is driven by the EVG and manipulated with software tool: EVGUI.
- TPGGUI is analogous to EVGUI.

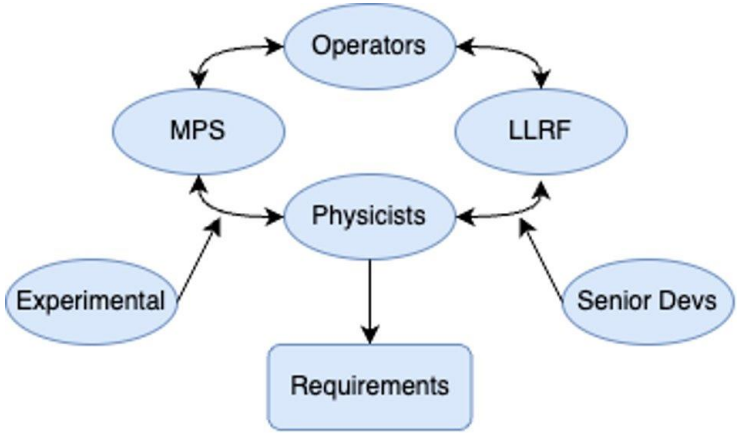


- Need to manipulate existing patterns and generate more complex patterns.
- Replaces manual construction/modification of requirements/parameters for existing patterns.
- Visually handle pulse assignment conflicts when scheduling beam to multiple destinations.

Requirements

- Code-specific Requirements:
  - Clean and maintainable code
  - UI to manipulate parameters
  - Reduce complexity for user/data visualization
- Machine Operation Requirements:
  - Keepalive triggers, BPM calibration triggers, BSA triggers, lower rate destination aware triggers
  - Maintaining Laser/Kicker thermal stability
- Coming soon: Formal Specification Document

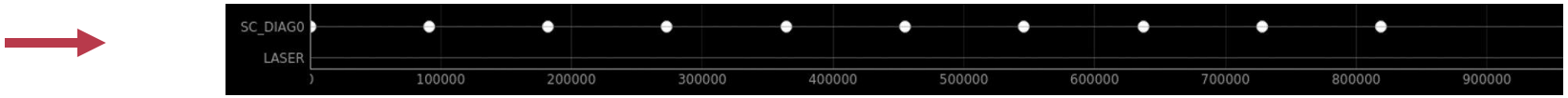
Stakeholders



What is a Timing Pattern?

10 Hz on the mark

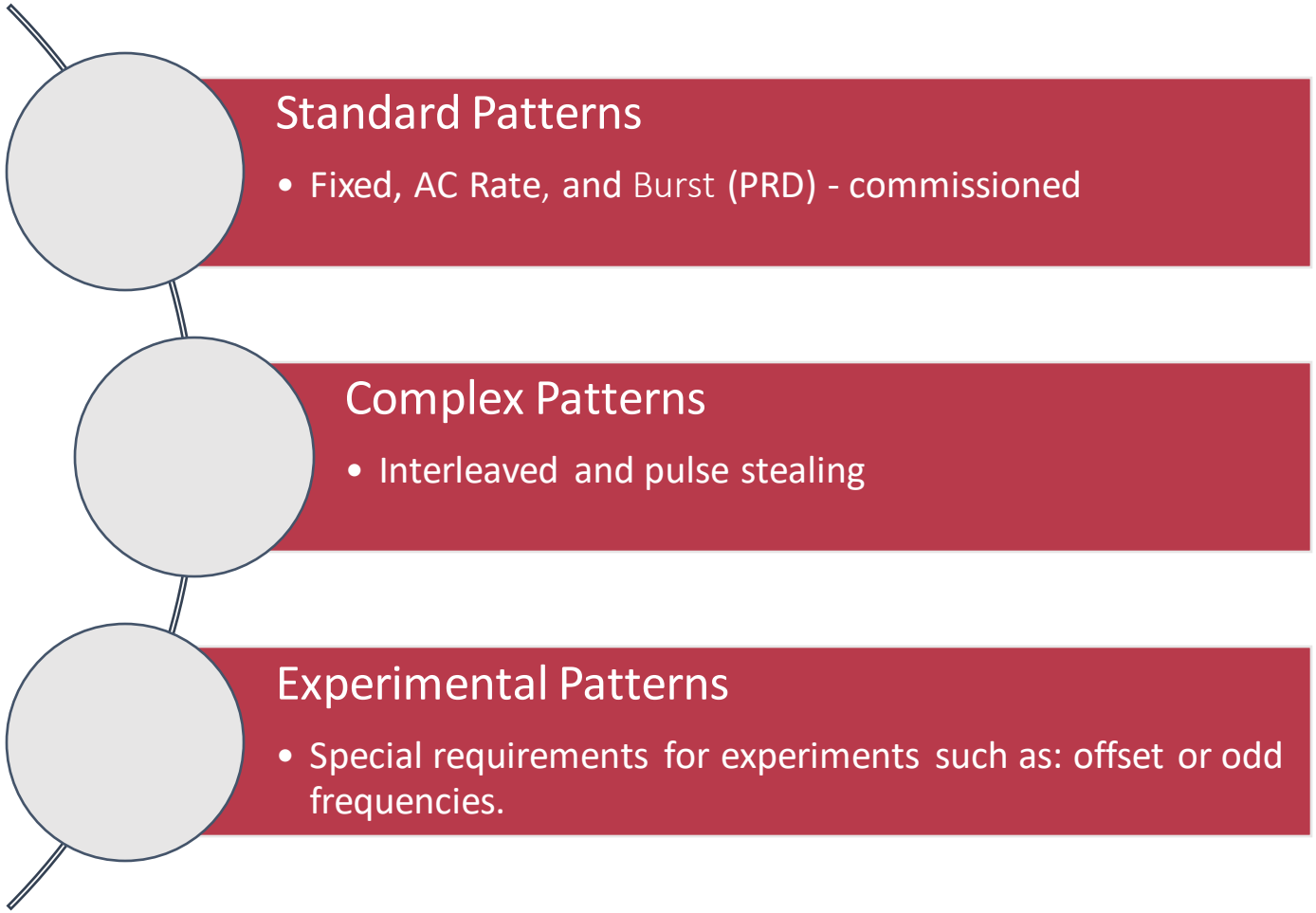
1. Send beam pulse
2. Wait for the 10 Hz marker
3. Go to instr. 1



- Set of instructions that tells the accelerator how, when and where to send a pulse

TPGGUI Pattern Categories

Timing Patterns Definition



Purpose for Categorization:

- "Protect" commissioned, established patterns: the Standard Patterns.
- Use the Standard Patterns as the base to re-define parameters and combine multiple of them to create additional complex patterns.

TPGGUI Project Goals

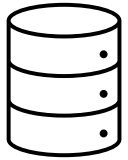
Generate Complex patterns and upload them to TPG for Operation

- Fixed Rate patterns interleaved within multiple destinations – offset one or more destinations from rate markers
- Fixed Rate patterns pulse-stealing within multiple destinations – destination priority
- AC patterns pulse-stealing within multiple destinations – destination priority
- AC and Fixed Rate patterns pulse-stealing within multiple destinations – destination priority
- Burst patterns for multiple destinations
- Burst and Fixed Rate patterns within multiple destinations
- Burst, AC Rate and Fixed Rate patterns within multiple destinations – destination priority
- Upload a complex pattern, slightly modify behavior and save as a new complex pattern

- Wishlist:

- Using the visualization tool to manipulate Burst patterns for multiple destinations
- Burst train of multiple burst patterns with different spacing

TPGGUI Data Flow



User Selected Inputs

- Rate Mode (AC/Fixed)
- Offset
- Repetition Rate (existing rate marker or desired rate)

Encoded Pattern Data

- Stored in directory containing multiple .json files

TPG updates the timing pattern

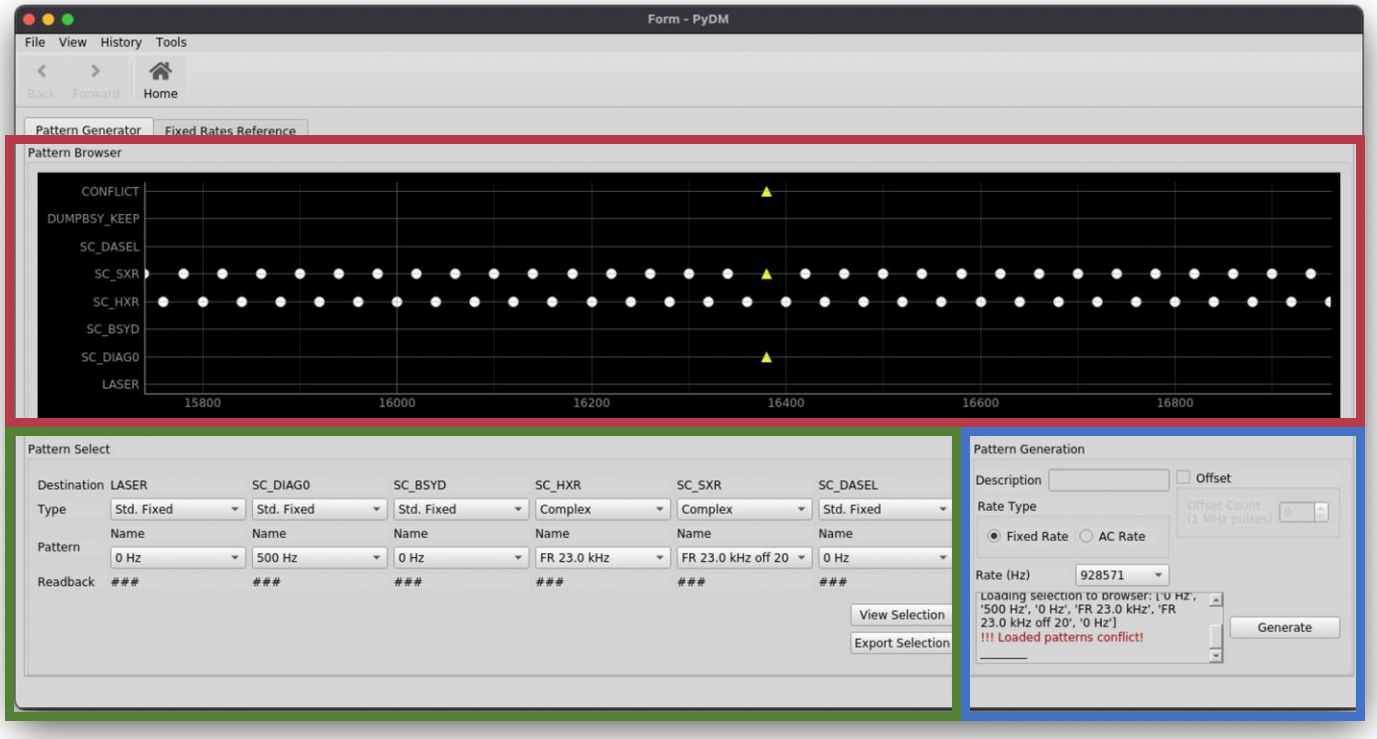
TPGGUI Codebase

- TPGGUI code is a python-based repo on git (clone with eco TPGGUI)
- The patterns are in JSON files organized in folders by destination
- The interface is PYDM-hybrid (.ui and .py files)

Current State of TPGGUI

1.

2.



3.

Sections

1. Pattern Browser: Visual display of selected patterns
2. Pattern Select: Destination by destination selection of patterns
3. Pattern Generation: Creation of new patterns

Live Demo

Roadmap

- Testing generated patterns in operation
- “Complex” Pattern Generator: used to generate patterns with multiple destinations
- Support changes to AC rates, burst patterns
- Synchronizing patterns to any rate markers

Acknowledgements

Carolina Bianchini Mattison

Matt Weaver

Mike Zelazny