RepRateDS instruction

12 October 2020 15:37

Attributes

inj_table_r1	String of R1 injection table currently in use
std_table_r1	String of R1 injection table read during device init
inj_table_r3	String of R3 injection table currently in use
std_table_r3	String of R3 injection table read during device init
ev_offset_r1	Clock cycles between injection event 0xa0 and chopper event 0xa4. Not currently used.
ev_offset_r3	Clock cycles between injection event 0xa0 and chopper event 0xa4. Not currently used.
event0_frequency_r1	Frequency of injection events 0xa0. The linac will be triggered at this frequency.
event0_frequency_r3	Frequency of injection events 0xa0. The linac will be triggered at this frequency.
event1_subfactor_r1	Sub factor of chopper event 0xa4 compared to 0xa0. Chopper event frequency will be f_event0 / subfactor
event1_subfactor_r3	Sub factor of chopper event 0xa4 compared to 0xa0. Chopper event frequency will be f_event0 / subfactor
event1_frequency_r1	Frequency of chopper events 0xa4. The chopper and kicker magnet will be triggered at this frequency.
event1_frequency_r3	Frequency of chopper events 0xa4. The chopper and kicker magnet will be triggered at this frequency.
current_ring_target	This selects which EVR is used to trigger the chopper DG645 delay generator via an RF switch.

Properties

evg_name_r1		Tango name for R1 event generator.
kicker_evr_name_r1	R1-D110210CAB04/TIM/EVR-01/Output00Delay	Tango full attribute name for R1 kicker event receiver.
chopper_evr_name_r1	I-K00/TIM/EVR-01-R1/Output01Delay	Tango full attribute name for R1 chopper event receiver.
evg_name_r3		Tango name for R3 event generator.
kicker_evr_name_r3	R3-A110111CAB04/TIM/EVR-01/Output00Delay	Tango full attribute name for R1 kicker event receiver.
chopper_evr_name_r3	I-K00/TIM/EVR-01-R3/Output01Delay	Tango full attribute name for R1 chopper event receiver.
n_max	10	Maximum number of entries in the injection table. Will limit the number of frequency settings that can be used.
t0	10e-9	Cycle time for the event clock.
event0_code	0xa0	Event code for the main linac injection event.
event1_code	0xa4	Event code for the secondary event triggering chopper and kicker.
data_pins	[37, 36]	The RaspberryPi used for running the device server controls RF switches with two GPIO pins. The naming of the pins is the BOARD convention, i.e. the pin number. The first pin is controlling the first switch, used to select the trigger source for the chopper delay generator.
ref_std_table_r1	"[[30, 160, 1]\n" "[18500, 161,	Stored reference table that can be written back to the EVG.

	1]\n " "[3, 162, 0]\n " "[4, 163, 0]\n " "[5, 164, 0]\n " "[6, 165, 0]\n " "[7, 166, 0]\n " "[8, 167, 0]\n " "[9, 168, 0]\n " "[10, 169, 0]]"	
ref_std_table_r3	"[[136, 160, 1]\n " "[10000, 161, 1]\n " "[3, 164, 0]\n " "[4, 160, 0]\n " "[5, 161, 0]\n " "[6, 160, 0]\n " "[7, 161, 0]\n " "[8, 164, 0]\n " "[9, 160, 0]\n " "[10, 161, 0]]"	Stored reference table that can be written back to the EVG.

Description

The dual reprate device server enables triggering the main linac and guns at one frequency and the chopper and kicker magnets at a lower frequency. The lower frequency is derived as a subfactor k of the main frequency. By doing this the thermal load of the structures in the linac can be kept constant at the higher frequency, e.g. 10 Hz, while electrons are only accelerated at the lower frequency, e.g. 2 Hz, to reduce radiation exposure.

This functionality is implemented by using two different events in the MRF system, one for triggering the main linac (event 0xa0) and the other for triggering the chopper and kicker (event 0xa4). The corresponding event receivers (EVR) are configured to listen to the relevant events and trigger various devices.

Injections are managed in the event generators (EVG) in the form of a sequence that is executed one entry at a time. Each entry in the sequence consists of an event code and a timestamp (clock cycle number). The EVG is clocked by the ring RF frequency (~100 MHz). When the internal clock counter reaches the next timestamp in the sequence, the event code for that position is emitted on the fiber network to the EVRs. The EVR has 12 outputs that can be individually configured to listen to a certain event and trigger after a set time delay. A special event code (0x7f) terminates and restarts the sequence. The sequence is implemented in the EVG device server as an event table with normally 10 entries stored as a device property and corresponding timestamp attributes.

The standard injection table consists of a 0xa0 event at a timestamp that aligns it with the desired RF bucket in the ring (changed from shot to shot) and some additional unused events followed by a 0x7f event at timestamp ~50000000 to give 2 Hz. When configuring for dual reprate the table is instead set to a 0xa0 event as before, immediately followed by a 0xa4 to trigger chopper and kicker. Then follows k-1 0xa0 events at the linac frequency and finally 0x7f at a timestamp that produces the desired lower frequency, e.g. 50000000 for 2 Hz.

Normally the DG645 delay generators that control the linac are triggered by a single EVR output and daisy chained so that one output of the last delay generator triggers the next. A selector box RF switch determines which EVR (SPF, R1, or R3) is used. This chain is interrupted for the dual rep rate triggering scheme by changing a cable so that the last DG645, which controls the chopper, is instead triggered by a separate output from an EVR. Another RF switch is then needed to select which EVR to use (R1 or R3).

The kickers are triggered directly from EVRs close to the rings so no cable modifications are needed there. The dual rep rate device server is run on a RaspberryPi4 in a 19" box in the timing cabinet where it also controls the RF switch via one of its GPIO outputs. Finally the EVR configuration is modified so that the outputs controlling the chopper and kicker listens to event 0xa4 instead of the normal 0xa0.

A peculiarity of the tango control system device servers is that device properties are not loaded onto the devices until an init command is executed. Both the injection table for the EVG and the event code for each output on the EVR are stored as properties. The init command is quite drastic in that it completely resets the device connections to hardware. However it appears that the EVG and EVR device servers can handle this without crashing. If it turns out that this is a problem the devices servers need to be rewritten to have properties changed to attributes instead.

The dual reprate device server has three injection tables stored internally for each ring. There is the reference table which is a device property and as such stored in the database and normally not modified. It can be set to recover the injection system to a known state. Then there is the standard table which is captured from EVGs during device init. This should be the latest used injection table. In principle it could be overwritten to a dual rep rate table if an init command is issued while a dual rep rate table is in place. Finally the dual rep rate tables are generated on the fly according to the set injection frequencies and subfactors. When generating a table it is not immediately set to the EVGs, a set_dual_reprate command must be issued first.

Usage

Generating new dual rep rate injection table

> Write one of the attributes `event0_frequency_rx` or `event1_factor_rx`

- A new injection table is generated internally but not written to the corresponding EVG.

Setting dual rep rate injection table

> Issue `command set_dual_reprate_table_to_rx`

- The injection is stopped (`inject_stop` command sent to EVG)
- The previously generated dual reprate table is written to the corresponding EVG. This done by writing the `sequence` property and eventxx_`timestamp` and eventxx_`enable` attributes.
- EVG is initialized by issuing `init` command.
- Chopper and kicker EVRs are configured to use the `event1_code` (`0xa4`) by writing the `output_init` property.
- EVRs are initialized by issuing `init` command
- Injection is still stopped and must be started again.

Setting standard injection table

> Issue `command set_current_standard_table_to_rx`

- The injection is stopped (`inject_stop` command sent to EVG)
- The standard injection table captured when initializing the device server is written to the corresponding EVG. This done by writing the `sequence` property and eventxx_`timestamp` and eventxx_`enable` attributes.
- EVG is initialized by issuing `init` command.
- Chopper and kicker EVRs are configured to use the `event0_code` (`0xa0`) by writing the `output_init` property.
- EVRs are initialized by issuing `init` command
- Injection is still stopped and must be started again.

Setting reference injection table

> Issue `command set_ref_table_to_rx`

- The injection is stopped (`inject_stop` command sent to EVG)
- The reference injection table stored in the property `ref_std_table_rx` is written to the

corresponding EVG. This done by writing the `sequence` property and eventxx`_timestamp` and eventxx`_enable` attributes.
- EVG is initialized by issuing `init` command.
- Chopper and kicker EVRs are configured to use the `event0_code` (`0xa0`) by writing the `output_init` property.
- EVRs are initialized by issuing `init` command
- Injection is still stopped and must be started again.

Switching rings without changing the injection table

> Write ring name `r1` or `r3` to `current_ring` attribute.

- The RF switch sets the corresponding ring's EVR to trigger the DG645 delay generator.
- `Event0_frequency_rx` reads the resulting linac injection frequency for dual rep rate injection table.
- `Event1_frequency_rx` reads the resulting kicker + chopper frequency for dual rep rate injection table. It should be `event0_frequency_rx / event1_subfactor_rx`.

Setting target ring RF bucket

> Write bucket timestamp to `event0_timestamp` in EVG

- This is the same as before.

Restore to standard configuration

- Connect cable from output xx on linac DG645 to chopper DG645 ext trigger.
- Set EVG injection tables to the standard ones.
- Set kicker EVR to listen to injection event (0xa0)