2604B, 2614B and 2634B – no external triggering

These lower price versions of the models do not have any digital IO for external triggering on the DB25 connector. Also, they have no TSP-LINK for triggering and communicating with other TSP instruments. And they have no contact check feature.

Last, the 2634B does not have the 100pA measure range that the corresponding 2636B model offers.

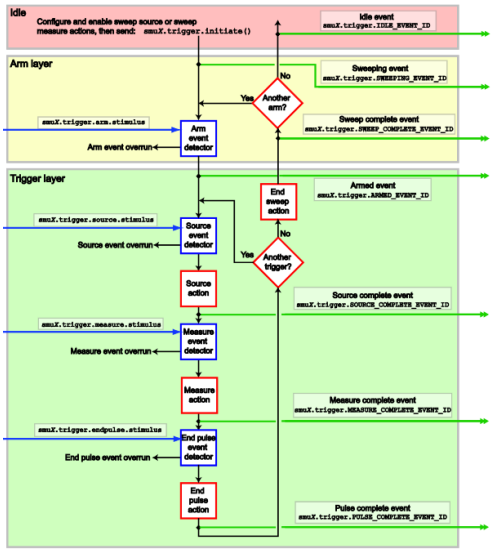
For coordinating the SMU with external equipment, the use of a software or bus-based triggering is still an option for the 2614B.

The trigger model diagram from the Reference Manual is copied below.

The Python sample code shows how the trigger model can be used to wait for the *TRG software trigger. When the smuX.trigger.initiate() command is sent, the SMU moves from the IDLE state and into the yellow Arm Layer. We've assigned the Arm Event Detector to use the trigger.EVENT_ID() as a stimulus which occurs when the *TRG is issued.

After the *TRG stimulus is received, the operation will advance into the green Trigger Layer and carry out the 7-point list sweep. The two entries in the source list will just be toggled thru, 7 times: 0, 5, 0, 5, 0, 5, 0

The point here is just to illustrate how the *TRG issued by the PC is an available means to coordinate SMU action with external devices.




```
my_instr.write("reset()")
my_instr.write("errorqueue.clear()")

my_instr.write("smu = node[1].smua")

my_instr.write("smu.nvbuffer1.clear()")
my_instr.write("smu.nvbuffer1.appendmode = 1")
my_instr.write("smu.nvbuffer2.clear()")
my_instr.write("smu.nvbuffer2.appendmode = 1")

my_instr.write("smu.source.func = smu.OUTPUT_DCVOLTS")
my_instr.write("smu.source.rangev = 5")
my_instr.write("smu.source.limiti = 0.01")

my_instr.write("smu.trigger.source.action = smu.ENABLE")
my_instr.write("smu.trigger.source.listv({0,5})")
my_instr.write("smu.trigger.source.limiti = 0.01")
my_instr.write("smu.trigger.measure.action = smu.ENABLE")
my_instr.write("smu.trigger.measure.iv(smu.nvbuffer1, smu.nvbuffer2)")
my_instr.write("smu.trigger.endpulse.action = smu.SOURCE_HOLD")
my_instr.write("smu.trigger.endsweep.action = smu.SOURCE_IDLE")
my_instr.write("smu.trigger.count = 7")
my_instr.write("smu.trigger.arm.stimulus = trigger.EVENT_ID")  # wait
for the *TRG command
my_instr.write("smu.trigger.source.stimulus = 0")
my_instr.write("smu.trigger.measure.stimulus = 0")
my_instr.write("smu.trigger.endpulse.stimulus = 0")

my_instr.write("smu.source.output = smu.OUTPUT_ON")
my_instr.write("smu.trigger.initiate()")

#after some delay the *TRG is sent
time.sleep(3)
my_instr.write("*TRG")
```