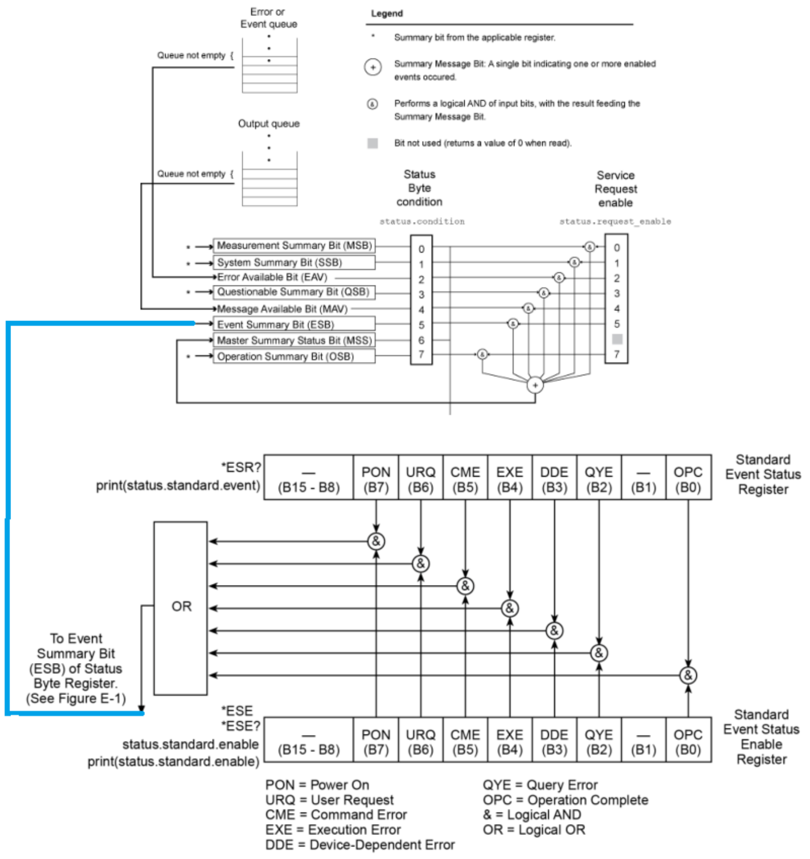
Using opc() to Coordinate Test Script Processing Instrument and PC/Python code

On the PC code, the read_stb() VISA command can obtain the value of the status byte on the instrument. The bits in this 8-bit register are controlled by configurations in other subordinate registers.

When the TSP code on the instrument executes the opc() command, this will cause bit 0 in the status.standard.event register to go to logic 1. If the corresponding bit in the status.standard.enable register is set, then this operation complete signal will flow through to status byte and bit 5 (ESB) will go to logic 1.
If the corresponding bit is set in the status.request_enable register, then bit 6 (MSS) will also go to logic 1 and an SRQ will be asserted.

The registers:

```
status.request_enable = 32   -- enable ESB to cause SRQ
status.standard.enable = 1   -- OPC bit in this register feeds the ESB bit
```


Error or Event queue
Queue not empty {
Output queue
Queue not empty {
Legend
* Summary bit from the applicable register.
Summary Message Bit: A single bit indicating one or more enabled events occured.
Performs a logical AND of input bits, with the result feeding the Summary Message Bit.
Bit not used (returns a value of 0 when read).
Status Byte condition
status.condition
Service Request enable
status.request_enable
Measurement Summary Bit (MSB)
System Summary Bit (SSB)
Error Available Bit (EAV)
Questionable Summary Bit (QSB)
Message Available Bit (MAV)
Event Summary Bit (ESB)
Master Summary Status Bit (MSS)
Operation Summary Bit (OSB)
*ESR?
print(status.standard.event)
— (B15 - B8)
PON (B7)
URQ (B6)
CME (B5)
EXE (B4)
DDE (B3)
QYE (B2)
— (B1)
OPC (B0)
Standard Event Status Register
OR
To Event Summary Bit (ESB) of Status Byte Register. (See Figure E-1)
*ESE
*ESE?
status.standard.enable
print(status.standard.enable)
Standard Event Status Enable Register
PON = Power On
URQ = User Request
CME = Command Error
EXE = Execution Error
DDE = Device-Dependent Error
QYE = Query Error
OPC = Operation Complete
& = Logical AND
OR = Logical OR

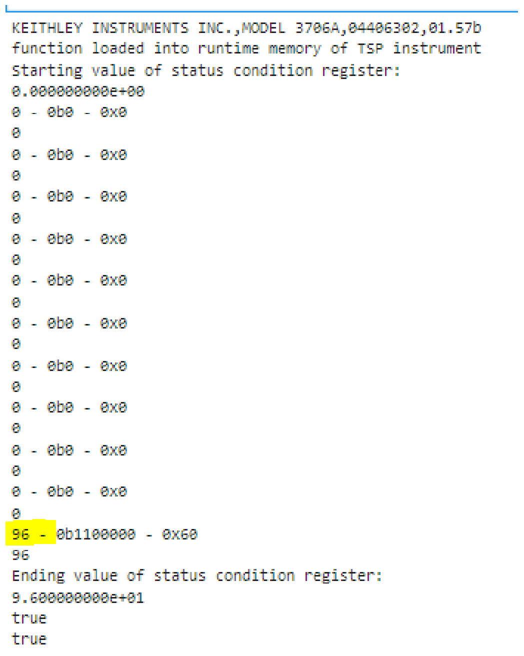
Python Code:

```
import pyvisa as visa
import time

rm = visa.ResourceManager()

try:
    #get a session/handle for one resource
    inst = rm.open_resource('TCPIP0::192.168.1.55::inst0::INSTR')
except(visa.VisaIOError):
    #try connect on dead socket on port 5030
    print("Attempting Dead Socket recovery.....")
    inst = rm.open_resource("TCPIP0::192.168.1.55::5030::SOCKET")
    inst.close()

    time.sleep(0.5)

    #try again now on normal LXI port

    inst = rm.open_resource('TCPIP0::192.168.1.55::inst0::INSTR')

    print("Done attempting Dead Socket recovery.....")



inst.write_termination = '\n'
inst.read_termination = '\n'
inst.send_end = True
inst.timeout = 10000   #timeout in msec

inst.write('*IDN?')
print(inst.read())

inst.write('reset()')
inst.write('errorqueue.clear()')
inst.write('status.reset()')

inst.write('status.request_enable = 32')
inst.write('status.standard.enable = 1')

inst.write("loadscript long_duration_task")
inst.write("function long_pause(duration)")
inst.write("delay(duration)")
inst.write("beeper.beep(0.5, 1200)")   # have the instrument give a chirp
inst.write("opc()")                    # have instrument tell us operation complete
inst.write("end")
inst.write("endscript")
#run the script to place into runtime memory of instrument
inst.write("long_duration_task.run()")
print("function loaded into runtime memory of TSP instrument")


print("Starting value of status condition register:")
print(inst.query('print(status.condition)'))

script_running = True
status_byte = 0
debug = 1
```

```
#call our long_pause function with a 10 second parameter
inst.write("long_pause(10)")


while script_running:
    status_byte = inst.read_stb()
    if debug: print(str(status_byte) + ' - ' + str(bin(status_byte)) + ' - ' + str(hex(status_byte))) # for debugging
    if debug: print(status_byte) # for debugging
    if (status_byte and 64) == 64:

        script_running = False

    time.sleep(1)   #delay for 1 second before asking again

print("Ending value of status condition register:")
print(inst.query('print(status.condition)'))
print(inst.query('print(bit.test(status.condition,6))'))  # bit.test index is one based, not 0
print(inst.query('print(bit.test(status.condition,7))'))

# close down
inst.clear()
inst.close()
rm.close()
```

Outputs from the Python:

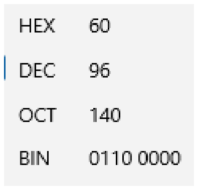
The simple delay task will take 10 seconds. We get new value of status byte every second from the loop. When bit 6 is logic 1, the while script_running loop will exit.



```
KEITHLEY INSTRUMENTS INC.,MODEL 3706A,04406302,01.57b
function loaded into runtime memory of TSP instrument
Starting value of status condition register:
0.000000000e+00
0 - 0b0 - 0x0
0
0 - 0b0 - 0x0
0
0 - 0b0 - 0x0
0
0 - 0b0 - 0x0
0
0 - 0b0 - 0x0
0
0 - 0b0 - 0x0
0
0 - 0b0 - 0x0
0
0 - 0b0 - 0x0
0
0 - 0b0 - 0x0
0
0 - 0b0 - 0x0
0
96 - 0b1100000 - 0x60
96
Ending value of status condition register:
9.600000000e+01
true
true
```

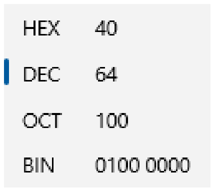
Decimal 96 = 0110 0000

Bits 5 and 6 at logic 1



HEX	60
DEC	96
OCT	140
BIN	0110 0000

Decimal 64 = bit 6 is logic 1



HEX	40
DEC	64
OCT	100
BIN	0100 0000