Cryogenic RF Switch Driver

- ✓ **An easy way to drive RF switches in low-temperature cryostats**
- ✓ **Tuneable switching pulses**
- ✓ **Remote SCPI interface**
- ✓ **Touchscreen interface**

Overview

The Cryogenic RF Switch Driver is a device to drive RF switches in cryogenic experiments. Switching pulses from a general purpose source can result in unwanted excessive heating of the experiment, resulting in a longer recovery time to reach the base temperature. The driver allows tuning the switching pulse properties to minimize the heating load. The outputs are two 8-pin connectors for easy wiring of RF switches with many ports. The driver offers a friendly local touchscreen GUI and a serial SCPI interface for remote operation.

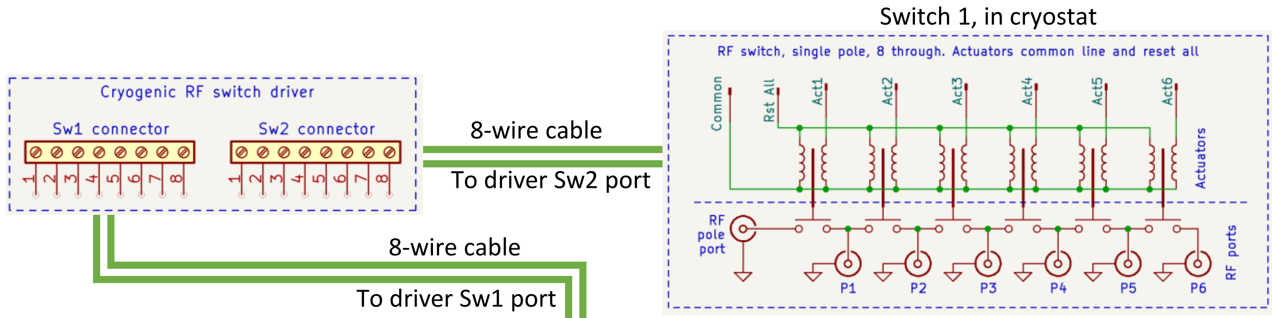
Applications

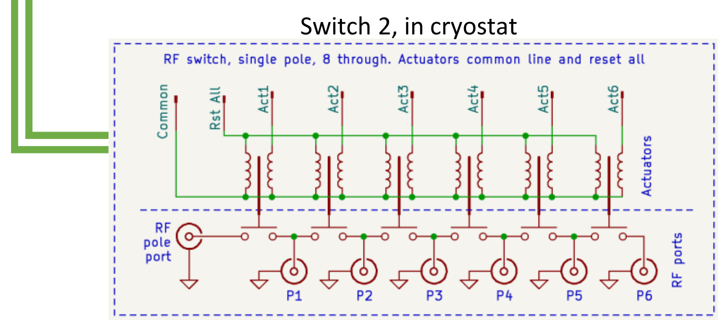
- Cryogenic RF calibration with thruline or SOLT
- Switching between different cryogenic experiments without warm-up

Features

- Two pulse output ports, 8-pin euro-style connectors
- Each port can drive a SP6T latching RF switch
- Pulses are generated by an internal current source
- Current direction is switched to set or reset an RF port
- Pulses can be tuned in length, current, rise and fall time
- Pulse current, range: 0mA – 750mA
- Pulse length, range: 1ms – 200ms
- Rise and fall time, range: 1ms – 8ms
- Gaussian rise and fall profile
- Pulse voltage compliance: 46V
- Required power supply: 48V, 50W minimum, with earth isolation
- Pulse lines are isolated from earth
- Pulse lines are isolated from device when not in use
- Touchscreen with user-friendly GUI
- Serial SCPI interface via USB port

Application example 1: switches with common line



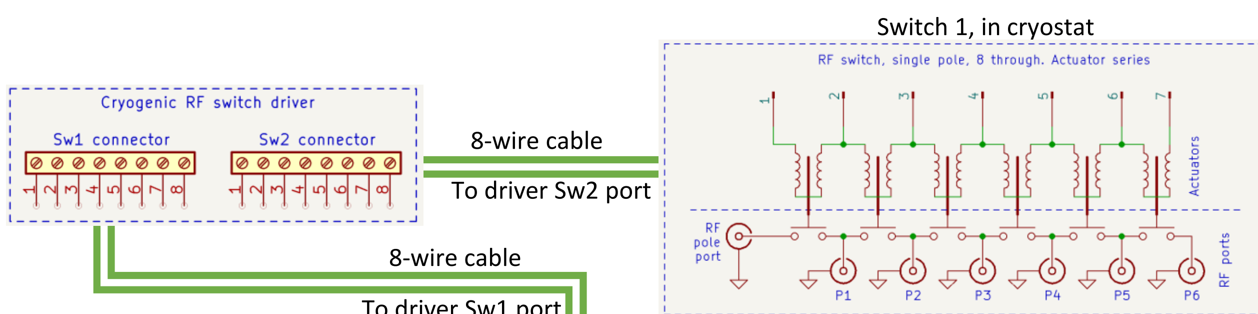


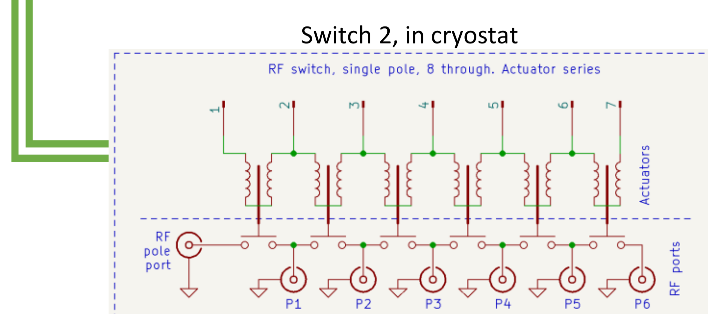
Drive Sw1 port	Switch 1
1	Common
2	Rst All
3	Act1
4	Act2
5	Act3
6	Act4
7	Act5
8	Act6

Drive Sw2 port	Switch 2
1	Common
2	Rst All
3	Act1
4	Act2
5	Act3
6	Act4
7	Act5
8	Act6

Port map

Application example 2: switches with actuator-series



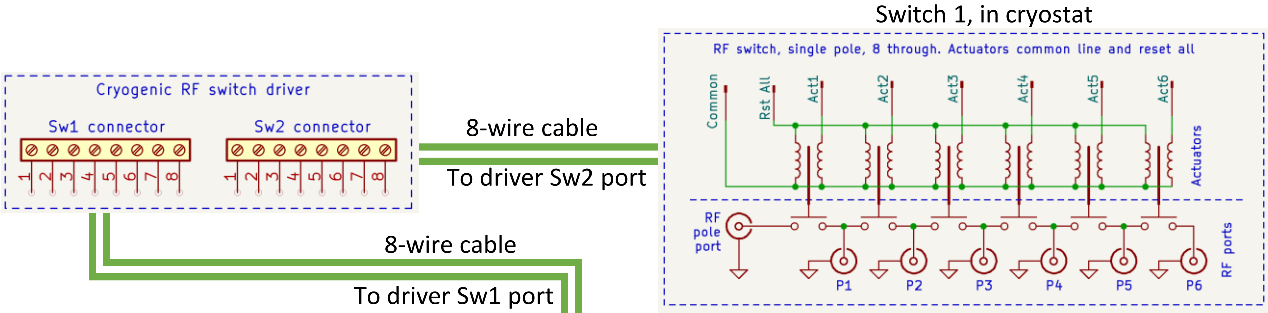


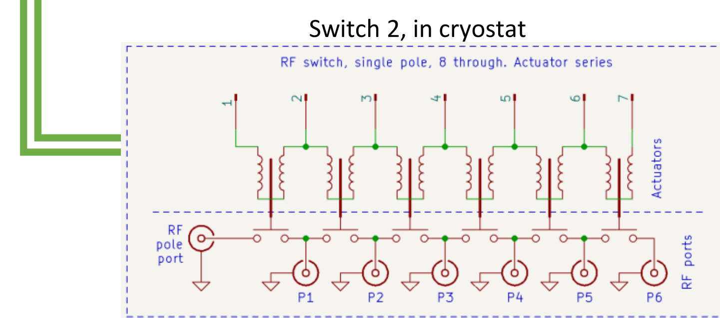
Driver, Sw1 port	Switch 1
1	Not conn
2	1
3	2
4	3
5	4
6	5
7	6
8	7

Driver, Sw2 port	Switch 2
1	Not conn
2	1
3	2
4	3
5	4
6	5
7	6
8	7

Port map

Application example 3: two switches of different type





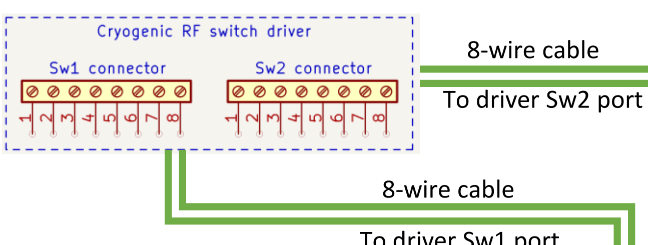
Drive Sw1 port	Switch 1
1	Common
2	Rst All
3	Act1
4	Act2
5	Act3
6	Act4
7	Act5
8	Act6

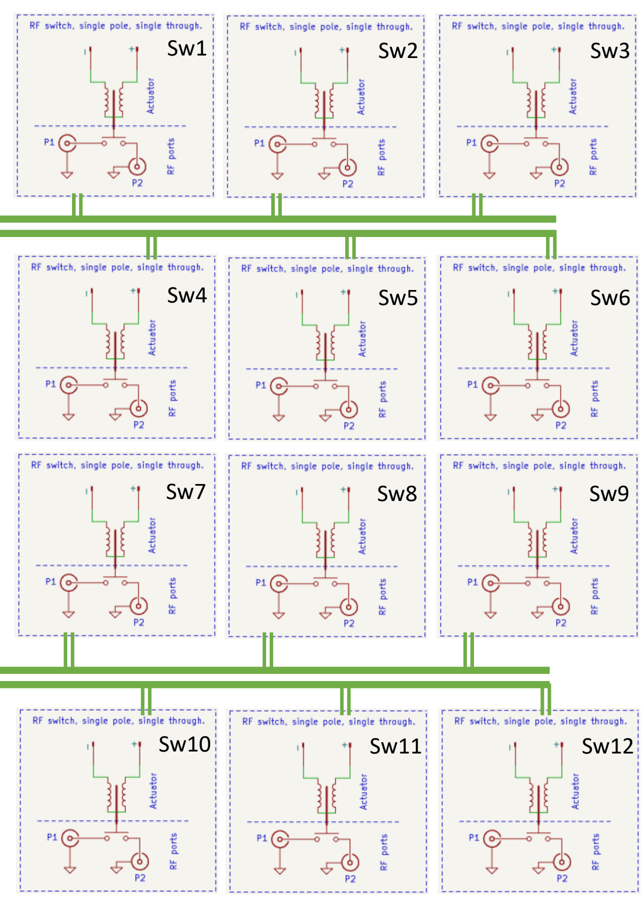
Driver, Sw2 port	Switch 2
1	Not conn
2	1
3	2
4	3
5	4
6	5
7	6
8	7

Port map

Application example 4: 12xSPST switches

This circuit is equivalent to application example 2, where the two SP6T switches are replaced by 6 SPST, and actuators are wired in series





Driver, Sw1 port	Switch
1	Not conn
2	Sw1 -
3	Sw1 + and Sw2 -
4	Sw2 + and Sw3 -
5	Sw3 + and Sw4 -
6	Sw4 + and Sw5 -
7	Sw5 + and Sw6 -
8	Sw6 +

Driver, Sw1 port	Switch
1	Not conn
2	Sw7 -
3	Sw7 + and Sw8 -
4	Sw8 + and Sw9 -
5	Sw9 + and Sw10 -
6	Sw10 + and Sw11 -
7	Sw11 + and Sw12 -
8	Sw12 +

Port map

SCPI interface use example with Python/Jupyter Notebook

In [1]:
```
import pyvisa
rm = pyvisa.ResourceManager()
import time
```

Open

In [2]:
```
rm.list_resources()
```
Out[2]: ('ASRL9::INSTR',)

In [3]:
```
RFSWDriver = rm.open_resource('ASRL9::INSTR')
RFSWDriver.write_termination = '\n'
RFSWDriver.read_termination = '\n'
RFSWDriver.baud_rate = 9600
```

In [4]:
```
RFSWDriver.query('*IDN?')
```
Out[4]: 'RF Switch driver\r'

Set / reset

In [5]:
```
# Set switch. Syntax: Switch (1 or 2), Port (1 to 6)
RFSWDriver.query('SET 1,1')
```
Out[5]: 'Sw1, P1 set.\r'

In [6]:
```
# Reset switch. Syntax: Switch (1 or 2), Port (1 to 6)
RFSWDriver.query('RST 1,1')
```
Out[6]: 'Sw1, P1 reset.\r'

In [7]:
```
# Reset All. Syntax: Switch (1 or 2)
RFSWDriver.query('RSTALL 2')
```
Out[7]: 'Sw2 reset all\r'

Switch status

In [16]:
```
# Port Status
RFSWDriver.query('STATUS')
```
Out[16]: 'SW1: P1:R || P2:S || P3:S || P4:S || P5:R || P6:R |||| SW2: P1:R || P2:R || P3:R || P4:S || P5:S || P6:S \r'

Read or set pulse parameters

In [17]:
```
# Set Rise Time. Syntax: Switch (1 or 2), Port (1 to 7) or 0, RiseTime ms (0 to 8).
#If Port=0, all the ports are set the same
#If Port=7, the setting is for Reset All
RFSWDriver.query('RISETIME 1,1, 2')
```
Out[17]: 'Rise Time updated\r'

In [18]:
```
# Read Rise Time. Syntax: Switch (1 or 2), Port (1 to 7)
RFSWDriver.query('RISETIME 1,1')
```
Out[18]: '2\r'

In [19]:
```
# Set Fall Time. Syntax: Switch (1 or 2), Port (1 to 7), Fall Time ms (0 to 8)
RFSWDriver.query('FALLTIME 1,1,5')
```
Out[19]: 'Fall Time updated\r'

In [20]:
```
# Read Fall Time. Syntax: Switch (1 or 2), Port (1 to 7)
RFSWDriver.query('FALLTIME 1,1')
```
Out[20]: '5\r'

In [21]:
```
# Set length. Syntax: Switch (1 or 2), Port (1 to 7), pulse length ms (0 to 1000)
RFSWDriver.query('LENGTH 1,1,20')
```
Out[21]: 'Pulse length updated\r'

In [22]:
```
# Read length. Syntax: Switch (1 or 2), Port (1 to 7), pulse length ms (0 to 1000)
RFSWDriver.query('LENGTH 1,1')
```
Out[22]: '20\r'

In [23]:
```
# Set  pulse current. Syntax: Switch (1 or 2), Port (1 to 7), pulse current mA (0 to 750)
RFSWDriver.query('CURR 1,1,400')
```
Out[23]: 'Pulse current updated\r'

In [24]:
```
# read pulse current. Syntax: Switch (1 or 2), Port (1 to 7)
RFSWDriver.query('CURR 1,1')
```
Out[24]: '400\r'

In [25]:
```
# Preset
RFSWDriver.query('PRESET')
```
Out[25]: 'Preset\r'

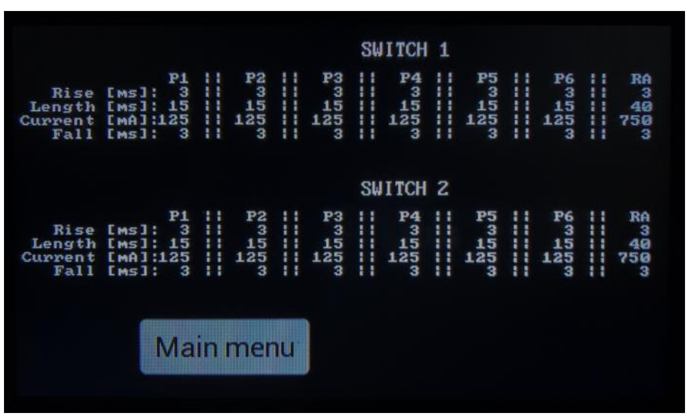
Pictures


Power and USB ports


Pulse ports, 2x8-pin euro-style

Main menu, with toggling buttons

Standby mode



Pulse library

Notes

- The driver can work only with latching relays
- The driver does not actively check the RF port status or the success of a switching event
- The port status displayed corresponds to the last pulse sent
- Pulse lines are isolated from earth only if an isolated power supply is used