Tutorial for the installation of Outer tracker hybrid test system

P. Chatagnon, F. Ferro, M. Gallo

January 8, 2025

Contents

1 Overview of the test system

This tutorial describes the steps to install the hybrid test system for the Phase 2 tracker of CMS. A schematic description of the test system is shown in Figure 1. The following instructions describe the installation of the connections between the PC and the multiplexing crate, the PC and the FC7 and the power supply of the crate.

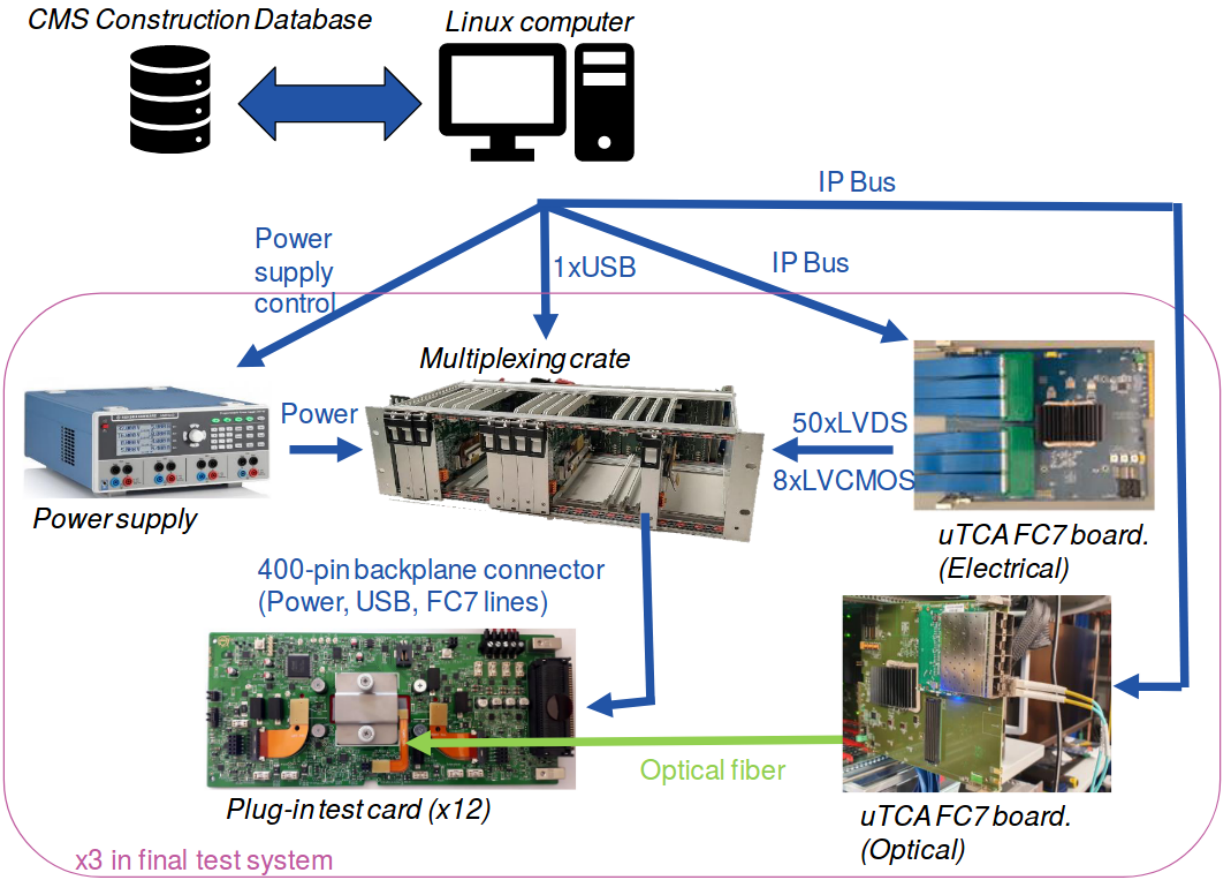


Figure 1: Scheme of the complete test system

Figure 2 shows the test system installed in Genova. The FC7 are hosted in the uTCA (top of the picture). The test crate is visible in the bottom of the picture.

2 Hardware

In this section the steps to install the various hardware parts of the system are detailed.

2.1 Power cabling and power supply setting

The test crate is powered by a *ROHDE&SCHARTZ HMP4040* power supply. The power cable is connected following the schematic of Figure 3. The various power cables are clearly labeled as shown in Figure 4.

Channel	CH1	CH2	CH3
Pair	P1V5	P3V3	M3V3
V (ROH)	1.8 V	3.5 V	3.6 V
V (POH)	10.5 V	3.5 V	3.6 V
A_{max}	2 A	4 A	1 A

Table 1: Voltage and current limit to apply to each channel of the power supply

The voltages and current limit have to be set as indicated in Table 1. Once the voltage and current are set, the cable can be plugged in the back of the crate in any power plug, as shown in Figure 5. The cable

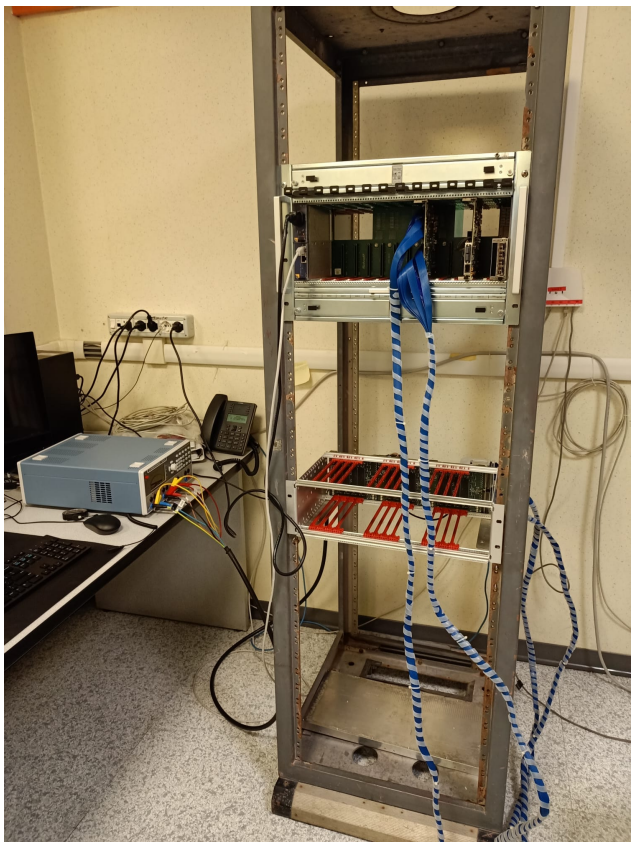


Figure 2: Overview of the test system in Genova

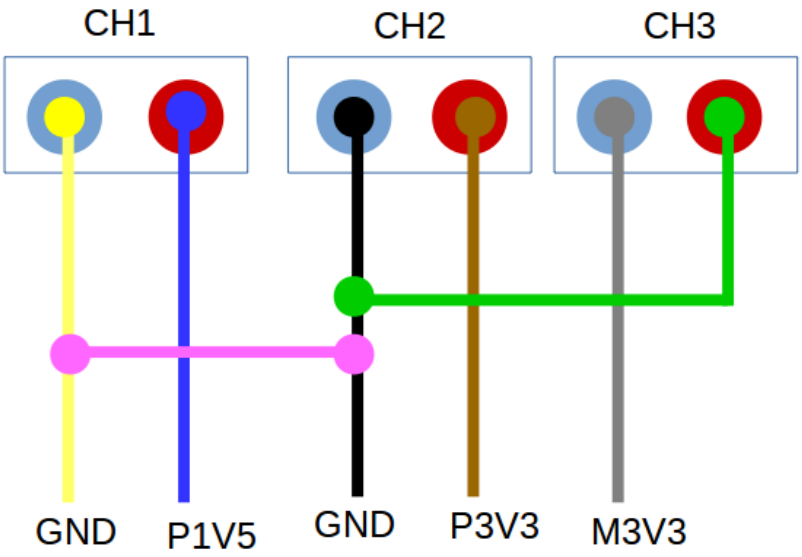


Figure 3: Schematic display of the connection of the crate power cable on the power supply.

connections for the test of ROH and POH are the same. Channel1 has a fixed voltage for ROH tests while for POH it becomes a *Test channel* and its voltage is remotely controlled. Note that, in the case of POH test, if the starting voltage of Channel1 is not set to 10.5V the test card cannot be configured.

The ethernet card of the PS must be correctly configured, proving a fixed IP address. It has to be in the same subnet as for the FC7's - see section 3.1 (*in our case: 192.168.0.10*).

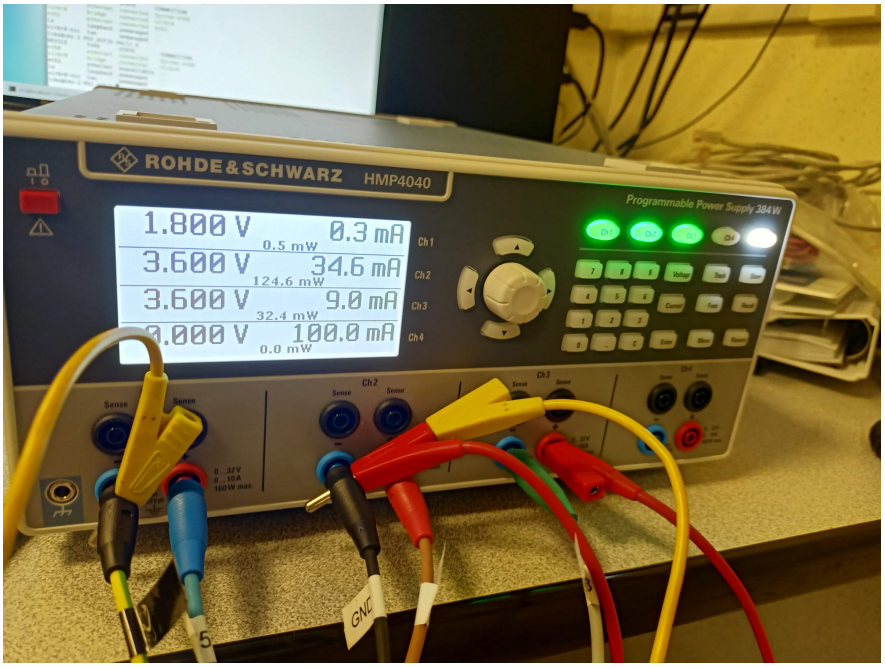


Figure 4: Fully configured power supply.

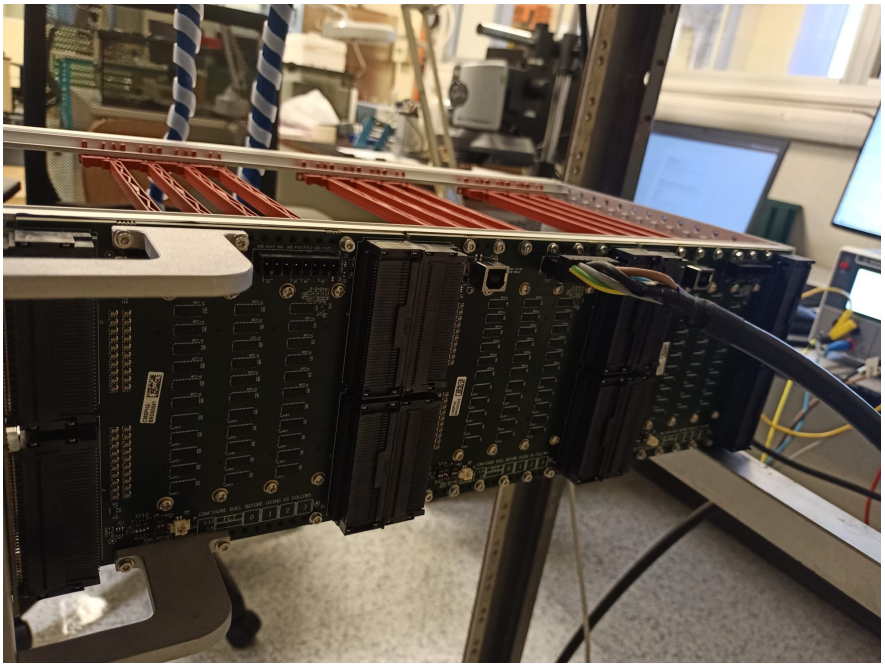


Figure 5: Picture showing the power cable connection to the crate. The cable can be plugged in any of the three backplanes (here it is on the middle one).

2.2 Backplanes setting

In order for the test card multiplexing to work, each backplane has to be given a unique adress. This is possible using physical switches located on the back of every backplanes. The switches are located as shown on Figure 6. Each switch corresponds to a unique address from 0 to 3 and is activated when in its upward position. The addressing of each backplane must be done in the following way: the backplane hosting the FMC connections must be addressed 0, the following one 1, and the furthest from the FMC connector 2.

2.3 Data transmission cables

Once the power cable is correctly installed, the FMCs and usb cables have to be installed.

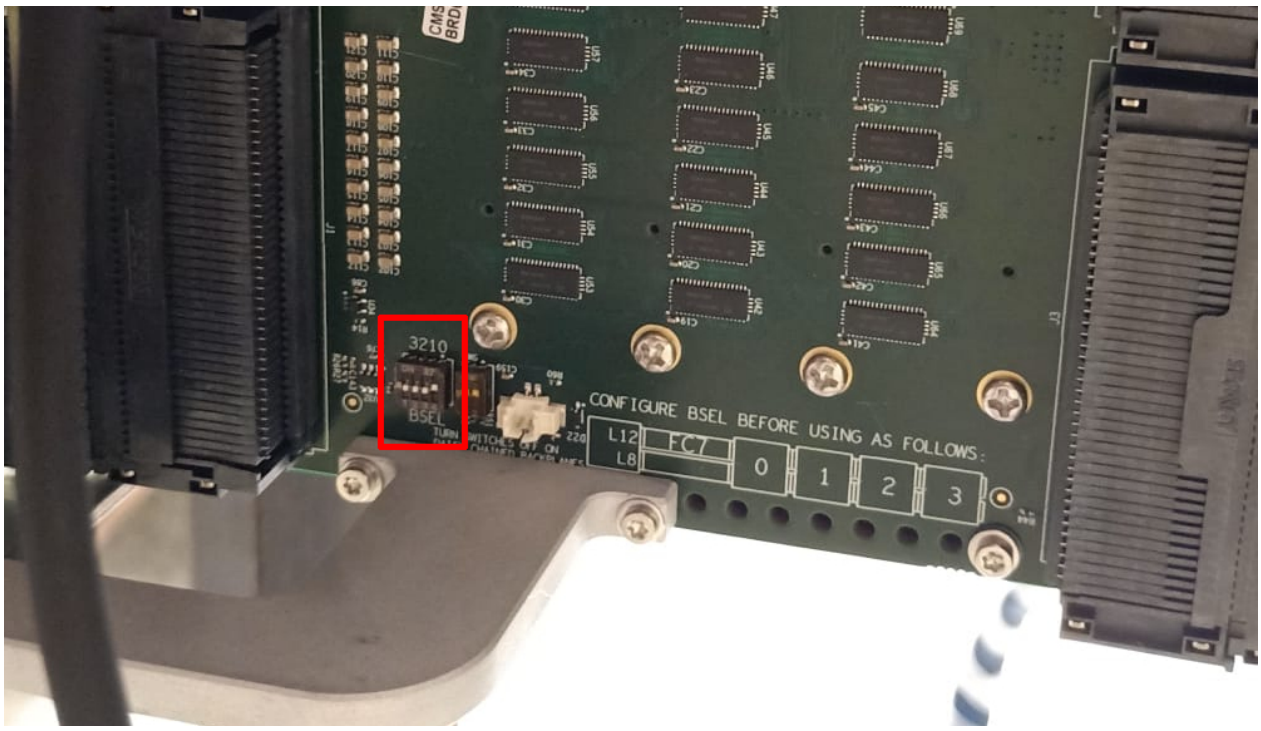


Figure 6: Switches used to provide a physical address to each backplane of a crate. The address corresponding to each switch is printed above each one of them, in this case the address is 0 corresponding to the backplane with the FMC connections to the FC7.

2.3.1 FMC cables connections

Two FMC cables link the test crate with the FC7 used to control the backplanes. The FC7 have two FMC connectors denoted L12 and L8, while the backplane connectors are denoted J1 and J2. The FMC cables have to be plugged as:

- L8 $\rightarrow$ J2
- L12 $\rightarrow$ J1.
- i.e. the upper part of the FC7 connected with the lower part of the mini-crate.

The blue flat canles should be connected to the backplane (bp) 0, as well as the USB cable.

Schematically the FMC connections are shown in Figure 7.

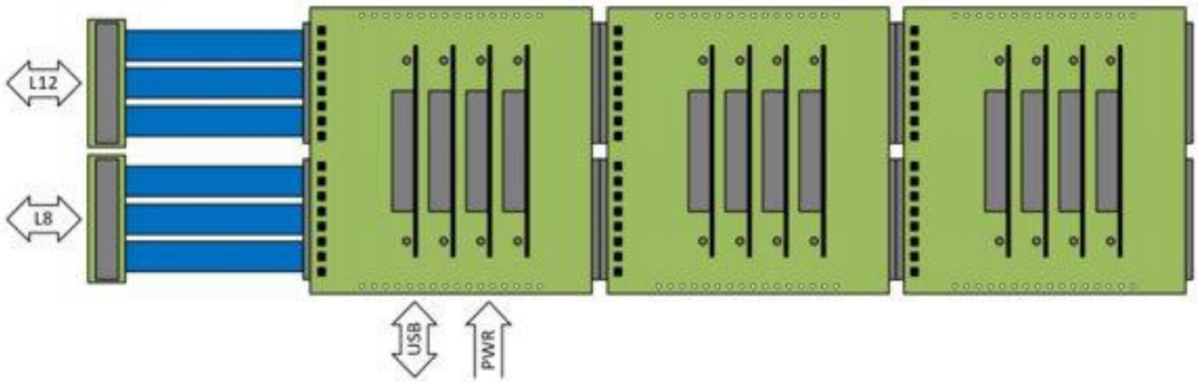


Figure 7: FMC Connection scheme

On both the FC7 and on the crate, the cables are secured using metallic pieces screwed in place. They can be seen on Figure 8 and 9. Finally, note that an adaptator is needed to plug the FMC cables on the crate. They can be seen in place on Figure 9.

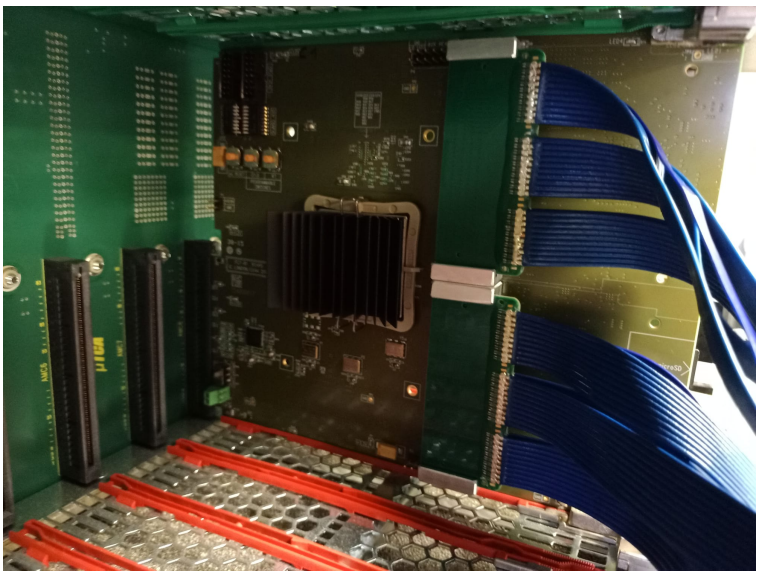


Figure 8: Picture of the FMC cable connection on the FC7. The connection is secured using rectangular metallic pieces, visible on the picture on each side of the connectors, which are screwed directly on the FC7.

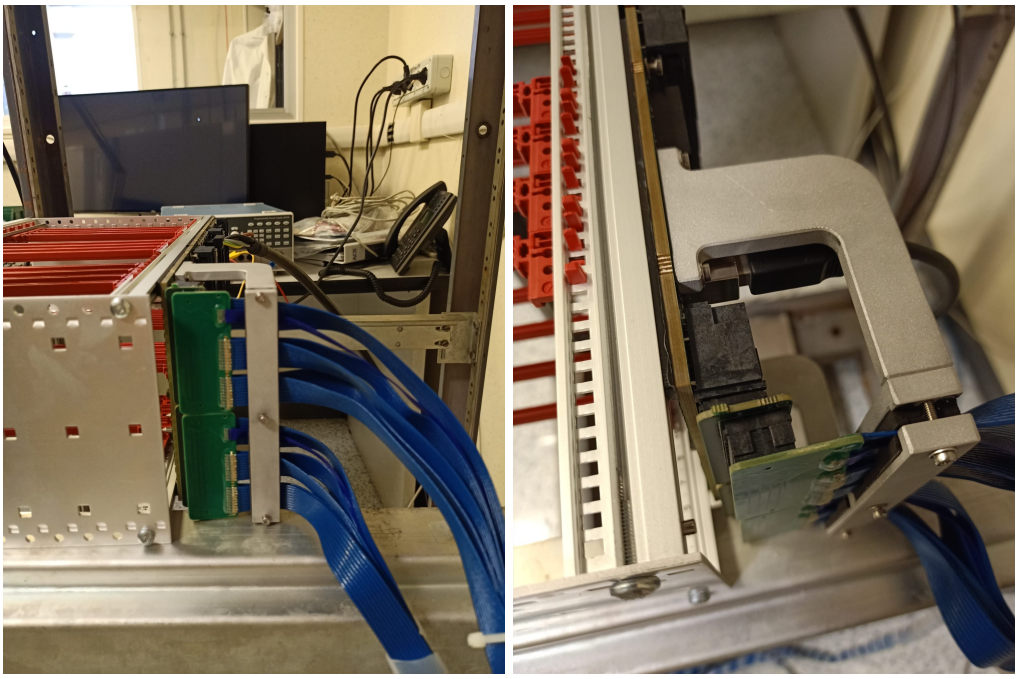


Figure 9: Picture of the connection of the FMC cable on the crate. The cables are secured using a long flat metalic piece with foam pad. The FMC adaptators are visible.

2.3.2 Usb cable connection

The USB connection between the PC and the crate is achieved using a USB-A/USB-B cable. The USB-B plug is connected in the back of the first backplane (i.e. the closest to the FMC connectors) as shown in Figure 10. Note that no usb device will be seen by the PC until a test card is correctly configured.

2.4 AMC13 module

In order to provide a clock to the system, the AMC13 module can be used (`http://www.amc13.info`). The module can be plugged as in Figure 11 and a loopback connection fiber must be plugged in. Note that even if the module is correctly powered no front panel led may be on.

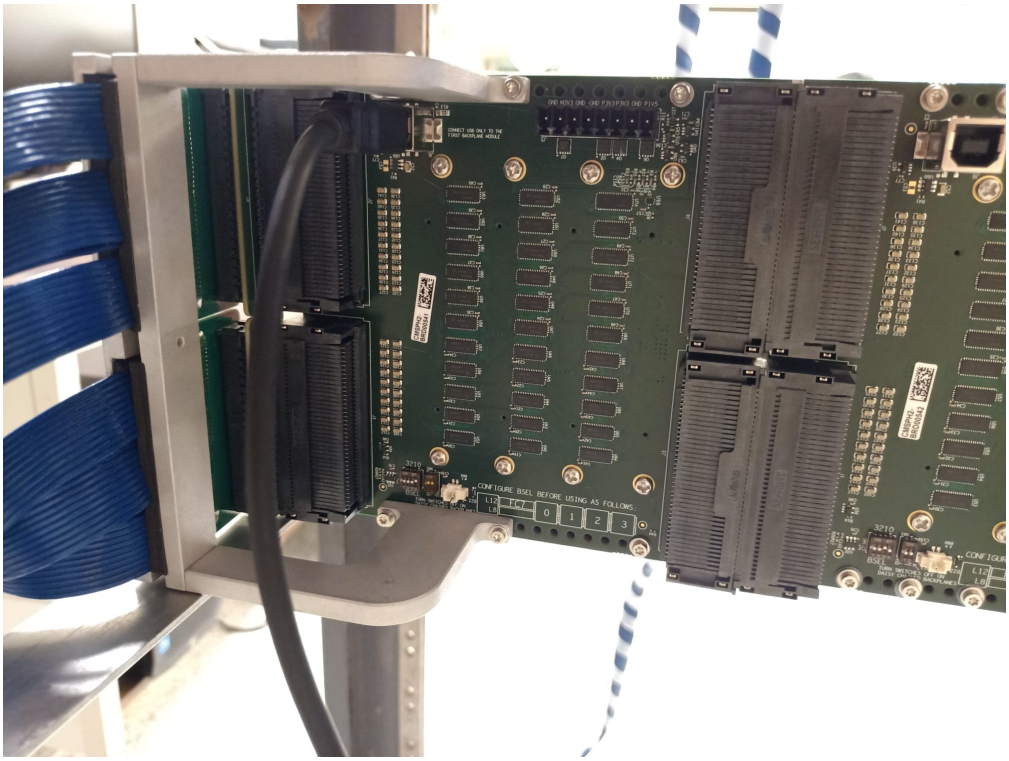


Figure 10: USB-B connection in the first backplane of the crate.

Figure 11: AMC13 module inside the crate.

3 PC setup and software installation

The following instructions complete the information and tutorials at:

- `https://cernbox.cern.ch/s/9snRcqXlD3EQW8w`
- `https://cms-tracker-daq.web.cern.ch/cms-tracker-daq/`

- https://cms-tracker-daq.web.cern.ch/cms-tracker-daq/tutorials/pc_connection/.

Make sure the PC you use has two ethernet cards (enp1s0 and enp2s0). We will set the connection on enp1s0.

A few more details on how to prepare the FC7 and SD cards can be found here: https://cernbox.cern.ch/s/stFuUdRMdO7cxF6 or alternatively https://www.ge.infn.it/~ferro/CMS/OTdocs/PreparingFC7-1.pdf

Before proceeding install a rarp daemon as explained in https://cms-tracker-daq.web.cern.ch/cms-tracker-daq/tutorials/pc_connection/.

3.1 Setting up the ethernet connection between the uTCA crate/FC7 and the PC

Once the rarp daemon has been installed and configured, do the following to verify the state of the enp1s0 interface and restart all the necessary services.

```
# Display the status of the ethernet interfaces of your PC
nmcli d

# Restart the various ethernet services
sudo systemctl daemon-reload
sudo systemctl restart rarpd
sudo systemctl restart NetworkManager
```

Set an IP adress for the enp1s0 interface

```
#Check the IP adress of the ethernet interface
ifconfig
#the inet field of enp1s0 where the uTCA crate is plugged should show an IP
adress, otherwise do the following line to provide an IP adress to enp1s0
sudo ifconfig enp1s0 192.168.0.3
```

The IP adress of the enp1s0 that we just configure as to be on the same network as the one of the FC7s. In other word with the enp1s0 adress set to 192.168.0.1, the IP adress of the FC7 have to be of the form 192.168.0.—

Also the subnet mask should be set to 255.255.0.0.

You can verify the state of the enp1s0 port using the following command:

```
# Verify that enp1s0 has state connected
nmcli d
```

If the ethernet port is correctly configured, the following output will be displayed:

```
DEVICE  TYPE      STATE                   CONNECTION
enp2s0  ethernet  collegato               enp2s0
enp1s0  ethernet  collegato               enp1s0
lo      loopback  connected (externally)  lo
```

Figure 12: Screen display of the *nmcli d* command if the enp1s0 port is well configured.

In case, a FC7 is configured in the uTCA crate, one can "ping" it to verify the ethernet connection.

```
# ping an FC_7 if it is already installed
ping 'Name of your FC7'
```

If the connection is established the output should look as in Figure 12, otherwise follow the instructions in the following section.

3.2 Adding a new FC7 to the uTCA crate

Switch off the uTCA crate, and plug the new FC7 in any available slot. Warning: to connect the FC7 it requieres a bit of force, make sure it is well plugged. If the following phase fails, first verify the FC7 connection.

Before switching on the crate, open a separate terminal and do:

```
tshark -i enp1s0
```

This command listen to the enp1s0 interface and display all the packets going through it. Now the uTCA crate on, look for the broadcast of the newly connect FC7. It provides its MAC adress. Note the MAC adress before proceeding.

```
#add a line in /etc/ethers with the MAC and a name for the FC7 (any you like)
sudo vim /etc/ethers

#add a line in /etc/hosts with a IP adress and the corresponding name
sudo vim /etc/hosts
```

The IP address of the new FC7 can be any address on the same sub-network as the ethernet port it is connected to: e.g. 192.168.0.— in our case. Obviously, it also has to be different from addresses already used in */etc/hosts*.

```
ping 'FC7 IP adress'
# or
ping 'FC7 name'
```

You should now be able to ping the new FC7.

```
[cms@cms-1 ~]$ ping fc7_ctl_1
PING fc7_ctl_1 (192.168.0.7) 56(84) bytes of data.
64 bytes from fc7_ctl_1 (192.168.0.7): icmp_seq=1 ttl=64 time=0.226 ms
64 bytes from fc7_ctl_1 (192.168.0.7): icmp_seq=2 ttl=64 time=0.210 ms
64 bytes from fc7_ctl_1 (192.168.0.7): icmp_seq=3 ttl=64 time=0.280 ms
64 bytes from fc7_ctl_1 (192.168.0.7): icmp_seq=4 ttl=64 time=0.206 ms
64 bytes from fc7_ctl_1 (192.168.0.7): icmp_seq=5 ttl=64 time=0.163 ms
```

Figure 13: Screen display of the ping command if the FC7 has been well configured.

Files */etc/hosts* and */etc/ethers* should look like in Fig. 14

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.0.3     enp1s0
192.168.0.100   mch
192.168.0.7     fc7_ctl_1
#192.168.0.111  FED
192.168.0.104   fc7_roh_1
192.168.0.13    AMC13_T2
192.168.0.12    AMC13_T1
/etc/hosts (END)
```

```
00:10:18:31:7b:18       enp1s0
00:40:42:0b:29:59       mch
#08:00:30:00:22:da      FED
08:00:30:00:29:89       fc7_ctl_1
#08:00:30:00:22:e0      fc7_ctl_1
08:00:30:00:29:2d       fc7_roh_1
08:00:30:f3:01:ac       AMC13_T2
08:00:30:f3:01:ec       AMC13_T1
/etc/ethers (END)
```

Figure 14: Screen display of /etc/hosts and /etc/ethers.

3.3 Changing MCH IP

For changing the MCH IP, one should run the command:

```
telnet mch
```

This command will open the MCH interface. Typing *h*, one can see the list of possible commands to execute. For now, the important one is *ip*. After running the IP command, one will see that it is possible to change the IP numbers, as shown below:

```
[cms@cms-1 ~]$ telnet mch
Trying 192.168.10.100...
Connected to mch.
Escape character is '^]'.


Welcome to NAT-MCH

nat> ip
IP Configuration Setup:
-----------------------
IP Address           : 192.168.10.100
```

Figure 15: MCH interface.

After changing the IP to the desired value, continue through the IP options, save the change and exit the MCH interface. Remember always to change to the right values in */etc/hosts* after the change.

3.4 Installing dependencies

Be sure to follow the README of every repositories to find and install other dependencies and prerequisites. (Especially the Ph2 might be more complex `https://ph2acf.docs.cern.ch/general/required_install/`) Access full instructions and links through Ref. 3.

For most packages, you can follow the usual instructions from each README, installing them in the same folder, here named as *test_hybrids*. Quite often it's just:

```
cd test_hybrids
git clone <package git url> --recurse-submodules
cd <package folder name>
. setup.sh
mkdir build
cd build
cmake3 ..
make -j8
```

The build order is as follows:

- cmsph2_tcusb (3.4.2)
- poh_tcusb (3.4.3)
- POWDER (3.4.4)
- tc_controller (3.4.5)
- Ph2_ACF (3.4.6)
- GUI_OTHYB (3.4.7)

After installing the required dependencies, one can also follow the steps to configure the power supply as a service (3.4.8) and the GUI application shortcut (3.4.9).

3.4.1 cmsph2_tcusb

Tcusb is a library handling the usb communication between the PC and the crate.
The installation has two parts, one for ROH and other for POH.

3.4.2 cmsph2_tcusb - ROH

The code can be downloaded from `https://gitlab.cern.ch/cms_tk_ph2/cmsph2_tcusb`.

3.4.3 cmsph2_tcusb - POH

The code can be downloaded from `https://gitlab.cern.ch/pszydlik/cmsph2_tcusb/-/tree/include_POH?ref_type=heads`. For this package, before doing the instructions explained in page 9, use `https://gitlab.cern.ch/pszydlik/cmsph2_tcusb/` for your git clone command, cloning into a different folder name than *cmsph2_tcusb* (such as *poh_tcusb*) and follow doing a checkout to the correct branch:

```
git clone https://gitlab.cern.ch/pszydlik/cmsph2_tcusb/ --recurse-submodules poh_tcusb
cd poh_tcusb
git checkout include_POH
```

3.4.4 POWDER

POWDER package must be installed to use the Power Supply from remote.
The code can be downloaded from `https://gitlab.cern.ch/cms_tk_ph2/power_supply`. After the installation, the config file, *power_supply/config/configRohdeSchwartz.xml*, must be properly modified:

```
File: power_supply/config/configRohdeSchwartz.xml

<?xml version="1.0"? encoding="UTF-8"?>
<Devices>
    <PowerSupply
        ID          =   "PSU"
        InUse       =   "Yes"
        Model       =   "RohdeSchwarz"
        Connection  =   "Ethernet"
        IPAddress   =   "192.168.0.10"
        Port        =   "5025"
    >
    <Channel ID="P3V3"          Channel="2" InUse ="No" />
    <Channel ID="N3V3"          Channel="3" InUse ="No" />
    <Channel ID="TestVoltage"   Channel="1" InUse ="Yes" />
    <Channel ID="Empty"         Channel="4" InUse ="No" />
    </PowerSupply>
</Devices>
```

Note that only the TestVoltage channel is remotely controlled. The other (2) channels must stay set as they are.

3.4.5 tc-controller

The code can be downloaded from `https://gitlab.cern.ch/cms-ot-hybrids/tc-controller.git` and installed following the instructions in the README.

A checkout to the *sync_with_usb_a* branch and cloning the *NetworkUtils* folder is necessary:

```
git clone https://gitlab.cern.ch/cms-ot-hybrids/tc-controller.git --recurse-submodules
cd tc-controller
git checkout sync_with_usb_a
rm -r NetworkUtils
git clone https://gitlab.cern.ch/cms_tk_ph2/NetworkUtils
```

After the installation, a config file must be added to define the test channel to be used by the controlled power supply. In our case, a file named *tc_controller/config/configGenova.xml* has to be added accordingly to the setup in POWDER package.

```
File: tc_controller/config/configGenova.xml

<PowerSupply ID="PSU">
    <TestVoltageChannel ID="TestVoltage"/>
</PowerSupply>
```

3.4.6 Ph2_ACF

Ph2_ACF is the main sodtware package for DAQ. The code can be downloaded from `https://gitlab.cern.ch/cms-ot-hybrids/Ph2_ACF_PS_ROH`. The README provides straight forward tutorial to compile the code. On Alma9, follow the instruction in the *"Setup on RHEL 9.1 or AlmaLinux 9.1"* section, and compile using the instruction in the *"The Ph2_ACF software"* section.

Be careful to install all the required packages. For what concerns *protobuf* package beware that during the checks (*make check*) a big amount of memory is allocated, thus one of the checks can fail simply bacause of not enough available RAM.

After the installation, change the file *Ph2_ACF_PS_ROH/settings/default.xml*, which the GUI's *configuration.json* is pointed to.

File: Ph2_ACF_PS_ROH/settings/default.xml

```
<?xml version="1.0" encoding="utf-8"?>
<HwDescription>
  <BeBoard Id="0" boardType="D19C" eventType="VR" linkReset="1" boardReset="1"
  configure="1">
    <connection id="board" uri="chtcp-2.0://localhost:10203?target=192.168.0.7:50001"
    address_table="file://settings/address_tables/uDTC_OT_address_table.xml" />
    <CDCE configure="0" clockRate="320"/>
    <OpticalGroup Id="0" FMCId="L8" reset="0"></OpticalGroup>
  </BeBoard>
</HwDescription>
```

Also, for Ph2 ROH seetings, *Ph2_ACF_PS_ROH/settings/PS_ROH.xml* needs to be changed to:

```
File: Ph2_ACF_PS_ROH/settings/PS_ROH.xml

<?xml version="1.0" encoding="utf-8"?>
<HwDescription>
  <!--###############
      OPTICAL FC7
  ################-->
  <!-- <BeBoard Id="0" boardType="D19C" eventType="VR"> -->
  <BeBoard Id="0" boardType="D19C" eventType="VR" linkReset="1" boardReset="1"
  configure="1">
      <connection id="board" uri="chtcp-2.0://localhost:10203?target=192.168.0.104:50001"
      address_table="file://settings/address_tables/uDTC_OT_address_table.xml" />
    <CDCE configure="0" clockRate="320"/>

<!--
    <OpticalGroup Id="0" FMCId="L8" reset="1">
        <lpGBT_Files path="${PH2ACF_BASE_DIR}/settings/lpGBTFiles/" />
        <lpGBT Id="0" version="1" optical="1" configFile="lpGBT_v1_PS.txt">
          <Settings
          />
        </lpGBT>
    </OpticalGroup>
-->
                                .
                                .
                                .
    <!-- 15 - default (internal oscillator), 0 - AMC13 -->
    <Register name="clock_source_u7">0</Register>
    <!-- 3 = 3 = internal oscillator (default), 1 = fmc_l8_clk1, 2 = fmc_l8_clk0,
    0 = COAX_IN -->
    <Register name="clock_source_u8">3</Register>
                                .
                                .
                                .
  <!--###############
    ELECTRICAL FC7
  ################-->
  <BeBoard Id="1" boardType="D19C" eventType="VR" linkReset="1" boardReset="1"
  configure="1">
    <connection id="board" uri="chtcp-2.0://localhost:10203?target=192.168.0.7:50001"
    address_table="file://settings/address_tables/uDTC_PSROH_elect_address_table.xml" />
    <CDCE configure="1" clockRate="320"/>


    <!--CONFIG-->
        <!-- 15 - default (internal oscillator), 0 - AMC13 -->
        <Register name="clock_source_u7">0</Register>
        <!-- 3 = internal oscillator (default), 1 = fmc_l8_clk1, 2 = fmc_l8_clk0,
        0 = COAX_IN -->
        <Register name="clock_source_u8">3</Register>


  </BeBoard>
```

3.4.7 GUI_OTHYB

The code can be downloaded from `https://gitlab.cern.ch/cms-ot-hybrids/gui-othyb.git`. This one, according to the README, it's just qt files compilation, no need for the procedure explained in page 9.

For the GUI and GUI dependencies:

```
git clone https://gitlab.cern.ch/cms-ot-hybrids/gui-othyb.git --recurse-submodules
cd gui-othyb
pip3 install -r requirements.txt
cd ui_files/
./compilePyQt5.sh
cd ../
```

To guarantee the success of database uploading, discussed further in section 3.7.1, make sure that all submodules inside GUI-OTHYB, py4DBupload, resthub and cmsdbldr, were installed properly.

gui-othyb/configuration.json file must also be updated. Paths and test command name must be correctly specified (configurations for ROH and POH tests are of course different).

The following parameters need to be changed carefully:

- default_Ph2_ACF_directory : the directory where Ph2_ACF is installed
- default_fw_image : the firmware image to be installed on the electrical FC7
- default_settings_files : a HW configuration xml file done on purpose where the electrical FC7 is board 0 (default.xml).

File: gui-othyb/configuration.json

```
{
    "default_Ph2_ACF_directory": "../Ph2_ACF_PS_ROH",
    "default_fw_image": "default_LV",
    "default_settings_files": [
        [
            "settings/default.xml"
        ]
    ],
    "default_serial_number_validator": "(((((PS)|(2S))-?(((FEH[0-9]{2}(R|L))|
    ROH[0-9]{2}|SEH|POH)))|(8CBC3((18)|(40))))-[0-9]{9})|(CMSPH2-PRT[0-9]{5})",
    "number_of_crates": 1,
    "number_of_hybrids": 1,
    "vendor_mode": true,
    "db_target": "production",
    "db_run_type": "VAT",
    "location": "Genova",
    "institute": "Genova",
    "tester": "Miguel Gallo",
    "hv_channels": [],
    "lv_channels": [
        [
            "PSU",
            "TestVoltage"
        ]
    ],
    "use_chamber": false,
    "climatic_chamber_IP": "192.168.1.40",
    "dwell_time": 600,
```

```
    "connect_to_othyb_db": false,
    "skip_upload": true,
    "skip_maint": true,
    "expert_mode": true
}
```

If the DB uploading is necessary or wanted, change the *skip_upload* flag to *false*.

The first time the GUI is used, the files *hybrid/PSROH.py* and *hybrid/PSPOH.py* needs to be updated. Especially the following variables:

- PH2_ACF_WORKING_DIRECTORY
- DESC_FILES
- FIRMWARE

For the file *gui-othyb/hybrids/PSPOH.py*, you should change those lines:

```
File: gui-othyb/hybrids/PSPOH.py

# CONFIGURATION VARIABLES
PH2_ACF_WORKING_DIRECTORY="../tc-controller"
DESC_FILES = [["./config/configGenova.xml"]]
SERIAL_NUMBER_VALIDATOR = "^PS(-)?POH-\d{9}$"
TEST_PROCEDURE = 'pspoh_w_root_file'
TEST_ARGUMENTS = [[ '--useLoadVector', '--tc_File',
'./config/configPSPOH_short.xml', '--supplyStep', '1', '--supplyMax',
'11', '--supplyMin', '8']]
FIRMWARE = [["2s_seh_electrical"]]
```

And, for *gui-othyb/hybrids/PSROH.py*, you should change those lines:

```
File: gui-othyb/hybrids/PSROH.py

# CONFIGURATION VARIABLES
PH2_ACF_WORKING_DIRECTORY="../Ph2_ACF_PS_ROH/"
DESC_FILES = [["settings/PS_ROH.xml"]]
SERIAL_NUMBER_VALIDATOR = ".*"
TEST_PROCEDURE = 'PSROHTest'
TEST_ARGUMENTS = [['--test-external-pattern', '0xAA', '--test-fcmd',
"--fcmd-pattern", "10", "--OptoEleClock", '--test-clock', '--test-reset',
'--test-i2c', '1000', '--measure-input-iv', '-b', '--thresholdFile',
'/home/cms/TrackerP2/OTHybrids/test_miguel_new/Ph2_ACF_PS_ROH/settings/
OTHybridTestingThresholdVars.yml']]
FIRMWARE = [["ps_12m_5g_cic2_l12octa_l8quad", "ps_12m_10g_cic2_l12octa_l8quad",
"ps_roh_5g_electrical", "ps_roh_10g_electrical"]]
#Crate 1: [Optical for 5G, Optical for 10G, Electrical for 5G, Electrical for 10G]
```

An additional change, with respect to the DB upload procedure, is also needed for all *gui-othyb/hybrids/*.py* files:

```
Files: gui-othyb/hybrids/*.py

db_loader = DBupload(database=BaseUploader.database,
login_type='login2', path_to_dbloader_api=db_loader_path,
verbose=True)
db_access = DBaccess(database=f'trker_{BaseUploader.database}',
login_type='login2', verbose=False)
```

This change will add a 2FA OTP login step for running the tests using the GUI, which is necessary for the DB upload procedure. More details at 3.7.1.

3.4.8 Power Supply as service

First make a script, which will be what the service runs continuously, */usr/local/bin/service_script.sh*:

```
File: /usr/local/bin/service\_script.sh

#!/bin/bash

cd test_hybrids/power_supply
. setup.sh
PowerSupplyController -c config/configRohdeSchwartz.xml --verbose
```

Make it executable

```
chmod u+x /usr/local/bin/service_script.sh
```

Then make a file to define the service, in */etc/systemd/system/powersupply.service*. The service name will be *powersupply*.

```
File: /etc/systemd/system/powersupply.service

[Unit]
Description=Power Supply Server service
After=network.target
StartLimitIntervalSec=0
[Service]
Type=simple
Restart=always
RestartSec=1
User=cms
ExecStart=/usr/local/bin/service_script.sh

[Install]
WantedBy=multi-user.target
```

Then reload the services and start the power supply service, using:

```
sudo systemctl daemon-reload
sudo systemctl start powersupply
```

To have the service start on startup:

```
sudo systemctl enable powersupply
```

3.4.9 GUI application shortcut

Couple steps are required to create a shortcut Icon for GUI execution to replace the manual run from commandline.

Create execution script in shared location i.e. */usr/local/share/GUI-HybTest.sh* which contains:

```
File: /usr/local/share/GUI-HybTest.sh

#!/bin/bash
cd /home/cms/TrackerP2/OTHybrids/test_hybrids/gui-othyb
. setup.sh
FILE_WITH_DATETIME=‘date +"GUI_LOG_%Y%m%d-%H%M%S.txt"‘
touch ./gui_exec_logs/${FILE_WITH_DATETIME}
python3 cratetesting.py &>gui_exec_logs/${FILE_WITH_DATETIME}
```

Make sure to create *gui-othyb/gui_exec_logs* directory in your system

Create desktop file in shared location i.e. */usr/local/share/applications/GUI_HybTest.desktop* which contains:

```
File: /usr/local/share/applications/GUI_HybTest.desktop

[Desktop Entry]
Type=Application
Terminal=true
Name=GUI-Hybrid-Testing
Icon=/home/cms/icon.png
Exec=/usr/local/share/GUI-HybTest.sh
```

Specify correct paths to exec script and to the chosen icon. Ensure that the shell script has correct user permissions so that you can run it. (you can test it by running it from terminal). One good option is to set chown and chmod to those two files:

Assuming you are using the cms user within the clean room PC:

```
sudo chown cms /usr/local/share/GUI-HybTest.sh
sudo chmod 755 /usr/local/share/GUI-HybTest.sh

sudo chown cms /usr/local/share/applications/GUI_HybTest.desktop
sudo chmod 755 /usr/local/share/applications/GUI_HybTest.desktop
```

And finally, execute the following command:

```
sudo update-desktop-database /usr/local/share/applications/
```

3.4.10 Updating packages with *git stash* command

One useful tool for the maintenance of the testing system is to know how to update any git cloned repository with the *git stash* command. It keeps all local changes not merged to the cloned branch, while updating for a newer version of the mentioned branch.

To stash the current changes, run:

```
git stash push -m "Description of the changes"
```

Before applying the stash, update the branch to incorporate the latest changes and reapply the stash:

```
git pull
git stash apply
```

If there are conflicts between the stash changes and the current branch, github will notify. Resolve the conflicts and then finalize the stash:

```
git stash drop
```

If the conflicts were not resolved and one proceed with the *git stash*, the command will keep both conflicted parts inside the script, indicating which part is related to the local and to the remote scripts. An example is showed below:

```
<<<<<<< Updated upstream
    DESC_FILES = [[
                    "../settings/PS_ROH_Top_12.xml",
                    "../settings/PS_ROH_Top_11.xml",
                    "../settings/PS_ROH_Top_10.xml",
                    "../settings/PS_ROH_Top_9.xml",
                    "../settings/PS_ROH_Top_8.xml",
                    "../settings/PS_ROH_Top_7.xml",
                    "../settings/PS_ROH_Top_6.xml",
                    "../settings/PS_ROH_Top_5.xml",
                    "../settings/PS_ROH_Top_4.xml",
                    "../settings/PS_ROH_Top_3.xml",
                    "../settings/PS_ROH_Top_2.xml",
                    "../settings/PS_ROH_Top_1.xml"
                ],
                ["../settings/PS_ROH_Mid.xml"],
                ["../settings/PS_ROH_Bot.xml"]]
=======
    DESC_FILES = [["settings/PS_ROH.xml"]]
    #             ["./settings/D19CDescription_PS.xml"],
    #             ["./settings/D19CDescription_PS.xml"]]
>>>>>>> Stashed changes
```

Figure 16: Git stash output example when there are a conflict script.

3.5 Communicating with the FC7/Crate

Never forget to set your command line to Ph2_ACF dir and run the setup script . *setup.sh* or *source setup.sh*.

3.5.1 Provide FC7 IP adress to the software

To communicate with the FC7 using PH2_ACF, you need to provide the IP adress of your FC7 to the program. This is done by changing the connection line in the xml description file (in our case *settings/PS_ROH.xml*) as:

```
<connection id="board" uri="chtcp-2.0://localhost:10203?
target='adress of your FC7':50001"
address_table="file://settings/address_tables/d19c_address_table.xml" />
```

and then do:

```
service controlhub start
```

Note that if you have more than one FC7 board you need to provide a different (0, 1, 2, ...) board Id for each FC7.

3.5.2 Writing the right firmware to the FC7 boards

Firmware of the OT tracker can be downloaded here: `https://udtc-ot-firmware.web.cern.ch/`. Cross-check with experts which is the latest version of the firmware for your HW configuration.

```
#Go in Ph2_ACF folder
cd Ph2_ACF #or Ph2_ACF_Latest or others depending on the latest installed version

#Display the available firmware on the SD card of the FC7
fpgaconfig -b 'boardId' -c settings/PS_ROH.xml -l

#Upload firmware from the PC to the FC7 SD card
fpgaconfig -b 'boardId' -c settings/PS_ROH.xml
-f 'firmware_file_name_on_the_PC.bin'
-i 'firmware_name_on_the_microSD'

#for the electrical FC7 (the one with the blue flat cables)
fpgaconfig -b 1 -c settings/PS_ROH.xml -f FC7_firmwares/ps_roh_5g_electrical.bin
-i ps_roh_5g_electrical

#for the optical FC7 (the one with the fibers)
fpgaconfig -b 0 -c settings/PS_ROH.xml -f ps_12m_5g_cic2_l12octa_l8quad.bin
-i ps_12m_5g_cic2_l12octa_l8quad

#Upload a firmware of the FC7 fpga
fpgaconfig -b 0 -c settings/PS_ROH.xml -i  ps_12m_5g_cic2_l12octa_l8quad
fpgaconfig -b 1 -c settings/PS_ROH.xml -i ps_roh_5g_electrical
```

In case of **POH** tests ONLY the electrical FC7 is used, since no data readout is needed. The boardId is in this case 0 (the HW settings can be just those in *settings/default.xml*) and there are two FWs that are used:

- default_LV.bin: used just to check for configurable cards in the mini-crate
- POH_HV.bin: used to run the test (TestVoltage must be set to 10.5V to have it working properly)

```
[cms@cms-1 Ph2_ACF]$ fpgaconfig -c settings/PS_ROH_el.xml -b 0 -i ps_12m_5g_cic1_l12octa_l8quad
14.07.2023 15:40:32: |140194246577280|I| Loading ps_12m_5g_cic1_l12octa_l8quad into the FPGA...
14.07.2023 15:40:35: |140194246577280|I| Firmware image: ps_12m_5g_cic1_l12octa_l8quad loaded on FPGA
14.07.2023 15:40:35: |140194246577280|I| >>> Done <<<
```

Figure 17: Screen display if the firmware has been correctly uploaded on the fpga

3.5.3 Communication with the mini-crate and test boards

First, a scan of the available test cards has to be done:

```
#Scan the crate, and display the slot occupied by a test card
mux_setup -f settings/PS_ROH.xml --mux_scan

#for POH
#mux_setup -f settings/default.xml --mux_scan
```

From the output (see figure) one can see how many cards are connected to each backplane in which position. In order to configure the connection one can run

```
29.08.2023 16:20:31: |140543984221312|I| Setup is scanned
29.08.2023 16:20:31: |140543984221312|I| Available cards for bp 0:[ 1 ]
29.08.2023 16:20:31: |140543984221312|I| Available cards for bp 1:[ No cards]
29.08.2023 16:20:31: |140543984221312|I| Available cards for bp 2:[ 2 ]
29.08.2023 16:20:31: |140543984221312|I| *** End of the operation ***
```

Figure 18: Screen display after a crate scanning. Two test cards are found.

```
#Configuring card 1 in backplane 0
mux_setup -f settings/PS_ROH.xml --mux_configure 0,1

#for POH
#mux_setup -f settings/default.xml --mux_configure 0,1
```

In order to setup the optical connection of the FC7 with the test card, one must identify the USB device via *lsusb* command looking for a line like

```
Bus 001 Device 005: ID 10c4:87a0 Silicon Labs PSRV2-CMSPH2-BRD00701
```

3.5.4 Using AMC13 clock

In order to have more reliable results, the clock provided by the AMC13 module should be used, instead the internal one. The AMC13 module should first set up and enabled. What follows has been extracted by the code and notes in gitlab repository at
`https://gitlab.cern.ch/cms-ot-hybrids/amc13-mch-configuration-scripts`.

First thing to do is to provide two IP adresses to the two tongues of the module T1 and T2. The procedure is the same as that used for adding a new FC7: use tshark to spot the MAC addresses of the two tongues and then edit /etc/ethers and /etc/hosts. The rule of thumb to identify T1 wrt T2 is that the one that keeps talking is T1. T1 and T2 IP addresses should differ by 1 digit (see Figure 14).

In order to be able to configure the AMC13, one can have a look at
`https://gitlab.cern.ch/cms-ot-hybrids/amc13-mch-configuration-scripts/-/blob/master/Getting_started_with_AMC13_module.pdf?ref_type=heads` (note that the instructions are for Centos7 and that some of the required packages may already be installed. Note also that the PATHs are sometimes referred

to the home ~ directory instead of the actual work dir. **Forget** about the IP assignment algorithm unless strictly needed for some reason).

After having installed the AMC13 code inside the amc13 dir, edit *connectionSN43.xml* with the right IPs and run

```
. env.sh
tools/bin/AMC13Tool2.exe -c amc13/etc/amc13/connectionSN43.xml
```

If the AMC13 is connected no error will be reported and with the *list* command it will be shown as in Figure 19. In order to configure the AMC13 the following line can be entered instead of the *list* command:

```
[cms@cms-1 amc13]$ tools/bin/AMC13Tool2.exe -c amc13/etc/amc13/connectionSN43.xml
No address table path specified.
Using .xml connection file ...
Using AMC13 software ver:0
Read firmware versions 0x224e 0x2e
flavor = 2  features = 0x000000b2
>list
Connected AMC13s
*0: SN: 211 T1v: 224e T2v: 002e cf: amc13/etc/amc13/connectionSN43.xml
>exit
```

Figure 19: Checking AMC13 connection.

```
> en 1-12 f t
> exit
```

The HW needs now to be instructed to use the clock provided by the AMC13 module. The file settings/PS_ROH.xml needs to be modified setting the clock_source_u7 register to 0 for each board:

```
<Register name="clock_source_u7">0</Register>
```

a new file can be created (settings/PS_ROH_AMC.xml) and it can be used to configure the boards and rerun the test.

3.6 Running the tests from command line

The use of the GUI is recommended to run the tests. In order to run the test from command line for ROH's:

```
#device number should be checked with lsusb
#to check connectivity
PSROHTest -f settings/PS_ROH.xml -b --USBBus 1 --USBDev 5

#run the full test
PSROHTest -f settings/PS_ROH.xml --ep 0xCA --test-fcmd --test-clock --test-i2c 1000 \
--test-eom --test-vtrx --test-reset --hybridId CMSPH2-PRT00193 -b \
--test-eom --measure-input-iv --USBBus 1 --USBDev 5
```

Note that all the optical channels are tested. At least one should be seen working. If all the channels fail there is a problem in the transmission/receiving of the optical signal.[1] [2] Note also that if channel 0 is

[1] FC7 version has to be checked.

[2] If an optical mezzanine is missing channel 0 might be seen as locked (!). Disable these channels to prevent errors.

enabled inside PS_ROH.xml, for some reason the test uses it instead of the locked channel and the test gets stuck.

In order to run the test from command line for POH's:

```
pspoh_w_root_file -f ./config/configGenova.xml --useLoadVector \
--tc_File ./config/configPSPOH_short.xml --supplyStep 1 --supplyMin 8 \
--supplyMax 11 --hybridId PSPOH-210000001 --output PSPOH-210000001 \
--USBBus 1  --USBDev 095
```

3.7 Running the tests using the GUI

CERN credential are requested the first time the GUI is executed (or the cache reset). This is done to access the DB. For the moment being only SSO accounts are supported (no two factor authentication) that are registered to one of the following CERN e-groups:

- Construction Operator: cms-tracker-assemblyOperators
- Quality Control Operator: cms-tracker-qcOperators
- Tracking Operator: cms-tracker-trackingOperators

The access is granted only to the *production* DB and not to the *development* one (see *configuration.json*).

To use the GUI, one has two options:

- Run the power supply as a service and execute the GUI application shortcut, named as "GUI-Hybrid-Testing"
- Open two terminal windows, one for the power supply and other for the GUI (see 3.7.1).

PS.: Always check if the Power Supply button "Output" is ON before running the tests.

3.7.1 DB upload procedure

For the 2FA OTP login, which is necessary to be done before running the GUI tests (when *"skip_upload": false*), run the following commands on a dedicated terminal, additional to the two mentioned before:

```
cd test_hybrids/gui-othyb
. py4dbupload/bin/setup.sh
. py4dbupload/bin/part_distribution.sh
```

When the last command is executed, one is going to be asked to fill the fields with one's CERN credentials (username and password) and the OTP code, which is generated by the same authenticator used for CERN related services.

If everything is well set and the command works, one should see something like this:

This procedure is necessary to be done daily, since the OTP expires after 24h.

3.7.2 GUI shortcut

Firstly, one have to check if the power supply service is running (or just restart it), using the following command:

```
sudo systemctl status powersupply
(sudo systemctl restart powersupply)
```

Then, one can click on the shortcut icon and proceed with the testing.

Figure 20: Print-screen when the credentials and OTP works as expected.

3.7.3 POWDER terminal

The power supply server must be started on a new terminal window and left running till needed

```
cd test_hybrids/power_supply
. setup.sh
PowerSupplyController -c config/configRohdeSchwartz.xml
```

3.7.4 GUI terminal

In a new terminal window other than the power supply one, run the following commands:

```
cd test_hybrids/gui-othyb
. setup.sh
python cratetesting.py
```

The GUI will ask how many hybrids are to be tested and to insert the full code (bar code) of each test card and hybrid module. No character will be typed if different from expected (e.g. usually names start with CMSPH2).
When the button to run the test is pushed, the GUI will first check for the presence of the boards in the mini-crate and then actually run the test.

If the test was executed with the terminals, one should expect a similar output to the print shown below when the test is finished.

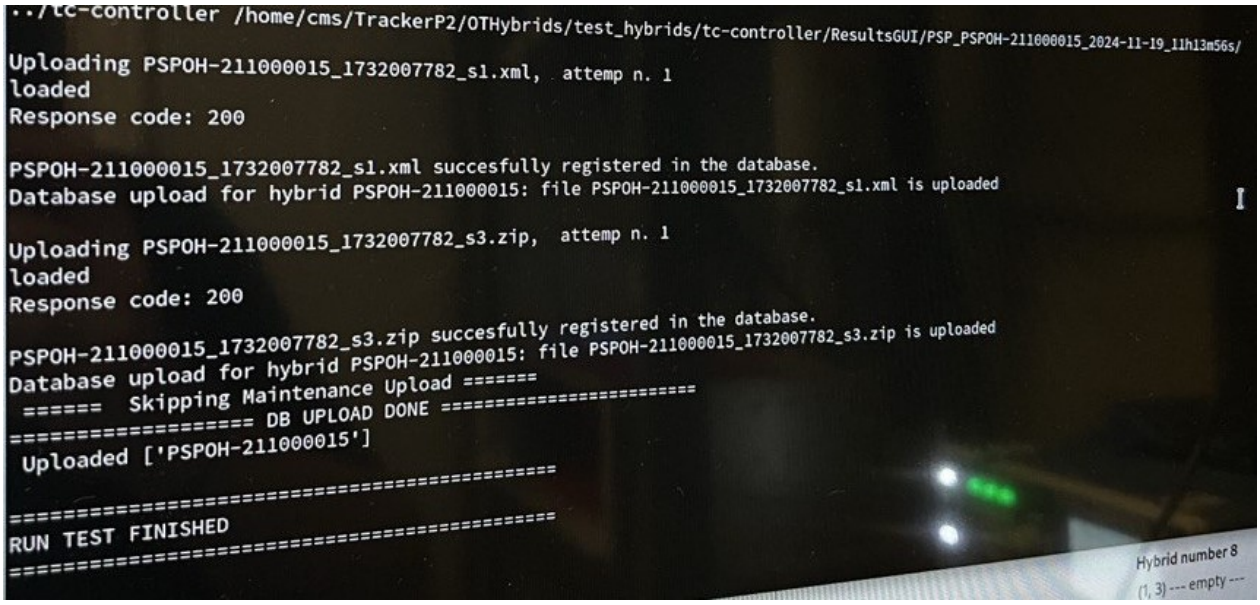


Figure 21: Print-screen when the DB upload is successfully done.

If the test was executed with the shortcut, the output is going to be saved in a file inside *gui-othyb/gui_exec_logs*.

Appendices

A Original instruction files

PS-POH

Tuesday, 7 March 2023 11:07

Instructions to run the PS-POH test on the cards.

To prepare the tests configuration:

1. Download and compile a specific cmsph2_tcusb repo with the File_refactor branch:
   Files · File_refactor · Patryk Szydlik / CMSPh2_TCUSB · GitLab (cern.ch)
   (It's not yet merged into new repository but it can freely replace the regular cmsph2_tcusb)
2. Remember to rebuild the Ph2_ACF after building cmsph2_tcusb
3. Download and build cms_tk_ph2 / POWDER · GitLab (cern.ch)
4. Configure your power supply in the *config/configRohdeSchwartz.xml*
5. *Make sure the power_supply server is running when executing tests*
6. Download and build CMS OT Hybrids / TC_Controller · GitLab (cern.ch)
7. Prepare a special config for your test voltage channel (examples of MiddleCrate included)

To select the card in the crate:

1. Set working directory:
   `/home/irene/Development/PSPOH/Ph2_ACF`
2. Set env variables with
   `source setup.sh`
3. Set the correct firmware
   'fpgaconfig -c settings/PSPOH.xml -i ps_poh_280922_final.bin'
4. Set correct voltage on the power supply (please don't burn anything)
5. Check available cards in the crate
   `mux_setup -f settings/PSPOH.xml --mux_scan`
6. Configure a card on backplane X, slot Y
   `mux_setup -f settings/PSPOH.xml --mux_configure X,Y`
7. Then you can check with `lsusb` that you can see the USB device:
   `Bus XXX Device YYY: ID 10c4:87a0 Silicon Labs`
8. To disconnect the selected card
   `mux_setup -f settings/PSPOH.xml --mux_disconnect`

To run the test procedure:

1. Change working directory to:
   `/home/irene/Development/PSPOH/TC-controller`
2. Set env variables with

`source setup.sh`

3. Select the test card using **Ph2_ACF**

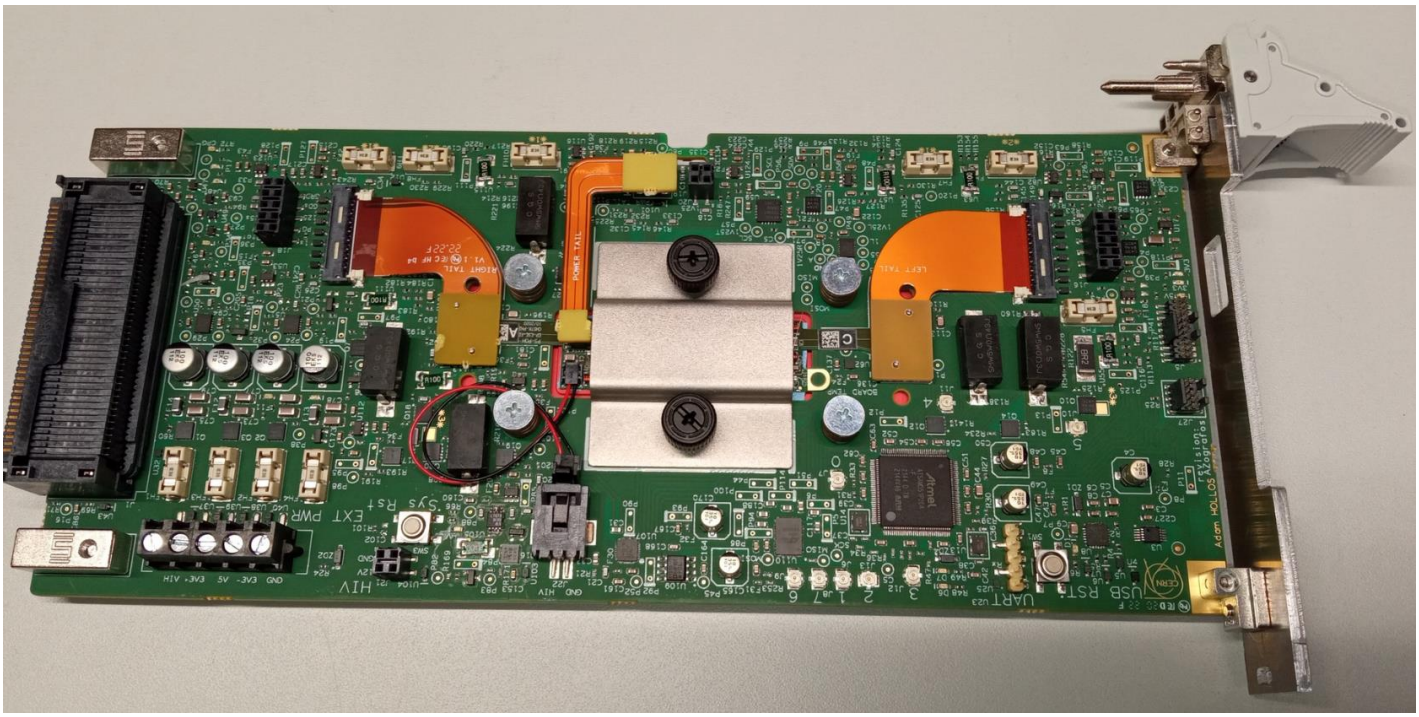
4. Find card USB bus (X) + dev (Y) numbers using **lsusb**. The USB device is the SiliconLabs one

5. Run the tests:
   ' pspoh_w_root_file -f ./config/configYourCrate.xml --useLoadVector --tc_File ./config/configPSPOH.xml --supplyStep 1 --supplyMin 8 --supplyMax 11 --hybridId <INSERT ID> --output <INSERT DIRNAME> --USBBus <INSERT BUS> --USBDev <INSERT DEV>'

PS-POH Test Card - Mounting of the hybrid

Tuesday, October 25, 2022

Authors: Farzad and Patryk

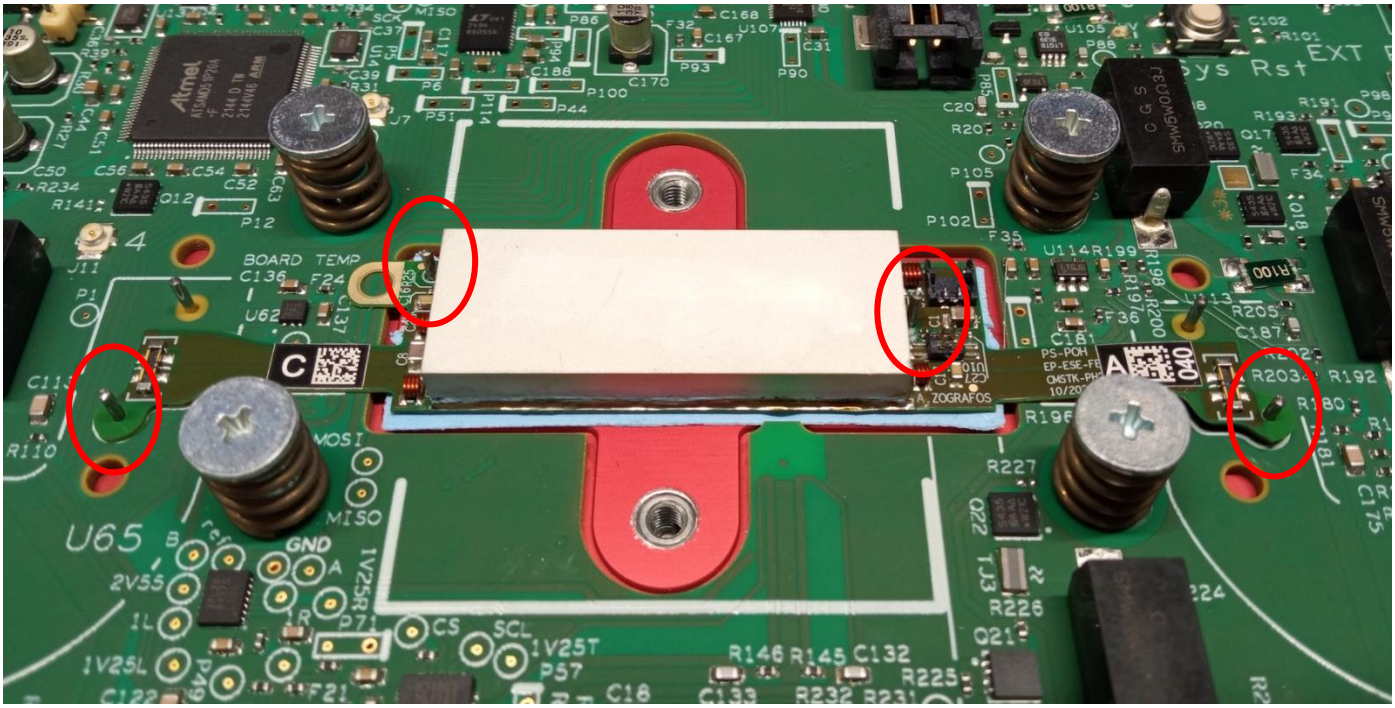


ALWAYS FOLLOW CLEANROOM AND ESD PRECAUTION!

- ground yourself with ESD strap,
- wear protective gloves.

Step 1: Mount the hybrid on the test card

Attention!
Align the holes on the hybrid with
The placement of alignment pins on testcard.

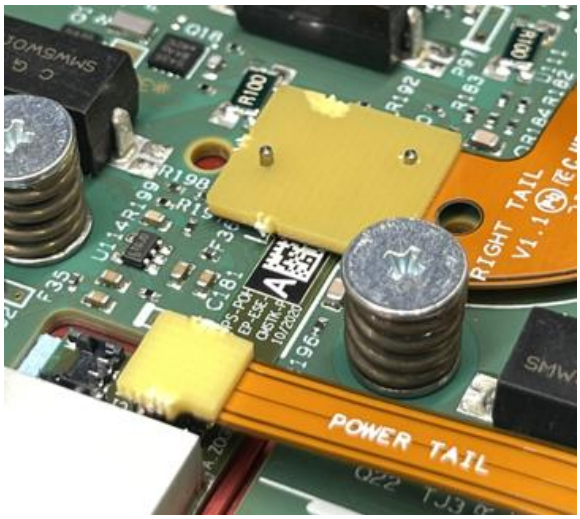


Step 2: Mate the tail jumpers

Attention!
Leave the connector socket on the test card open
for this process to allow for small movements.

Step 3: Plug in the power jumpers to the hybrid

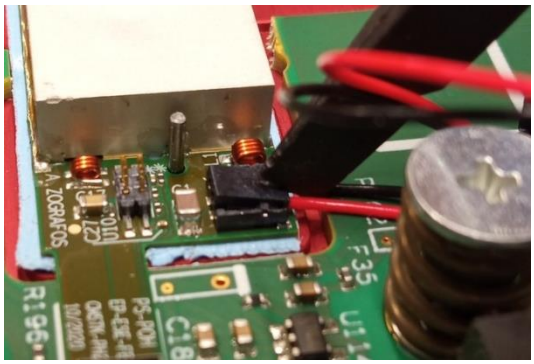
Power tail:

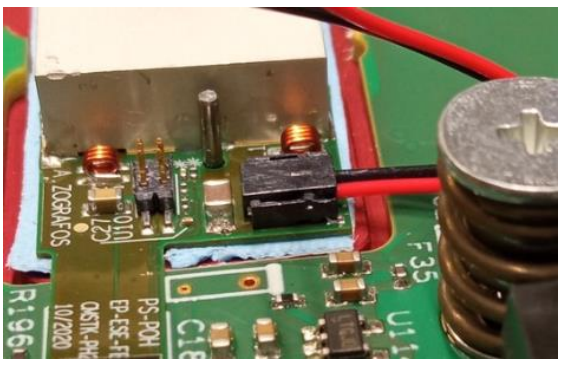


Power cable:

Attention!

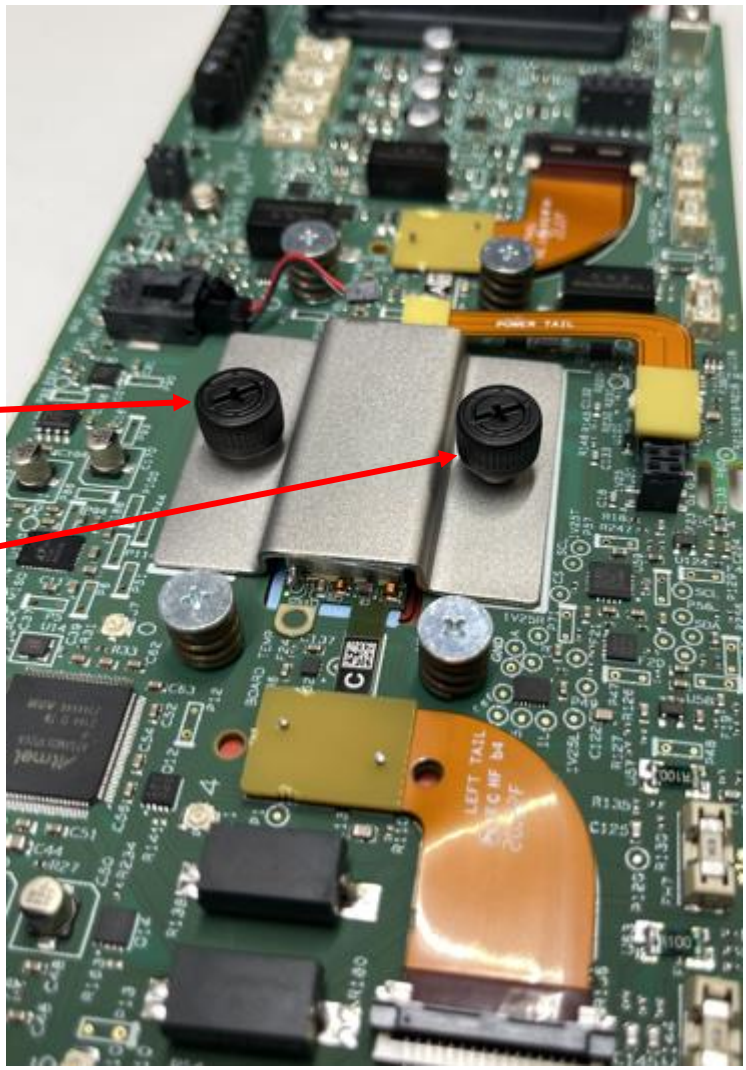
According to mating instruction of the connector manufacturer it should first aligned on the bottom of the connector and stick to the socket and afterwards fully mated by pressing top part of connector.





Step 4: Mount the top bracket on the test card over the hybrid.

Attention!
Assembled cards contain a stopper screw, it is impossible to press on the hybrid excessively, you can freely turn the screws until they reach the end.



Step 5: Close the tail connectors mounting.

Parts