

SPICA5 FW and API interwork

Spica5 SDK team

13 May 2022



Spica5 FW & API interwork

Spica5 SDK team

04 May 2022

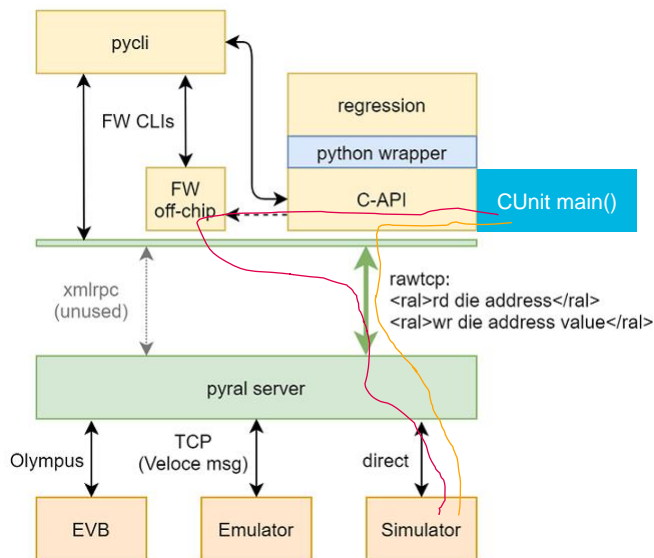


Spica5 Overview

API Unit test:

check unittest c code – api – fw – simulator debug
(in dc5 chamber: <https://dc5lp-vinetx000.marvell.com/>)

CUnit Test Models



- Start pyral along with simulator:

```
- [venv] [dile@dc5lp-vinex076 ~/python3_venv]$ python ${HSC_DIR}/spica5/tools/pyral/scripts/ral_server --chip spica5 --port 50000 simulator
```

- Start off-chip FW:

```
- [venv] [dile@dc5lp-vinex076 ~/python3_venv]$ ${HSC_DIR}/spica5/firmware/application/x86/spica5_fw_app --ip localhost --port 50000 --cli-port 52000
```

- Either start unit test:

```
- ${HSC_DIR}/spica5/api/capi/build-output/tests/tests --rawtcp --ip 127.0.0.1 --port 50000 --product spica5
```

- Or start gdb to run the unit test

Ref: overview.md

Setup ral_server, simulator and x86 FW

- Start the ral_server along with the simulator in python3 environment:

- [venv] [dle@dc5lp-vinetx076 ~/python3_venv]\$ python \${HSC_DIR}/spica5/tools/pyral/scripts/ral_server --chip spica5 --port 50000 simulator

- Compile the FW in x86:

- ```
cd ${HSC_DIR}/spica5/firmware/application
```

- ```
# Debug mode:
```

- ```
We invoke Meson with the setup command, giving it the location of the build directory. Meson uses out of source builds.
```

- ```
# Meson setup just need to be done once
```

- ```
meson setup .x86 --buildtype=plain
```

- ```
meson setup .x86 -Dxmlrpc=false --reconfigure
```

- ```
To start the build, simply type the following command.
```

- ```
# Meson will use the Ninja backend to build project
```

- ```
meson compile -C .x86 -j 3
```

- ```
# If you want to clean, following command can be used.
```

- ```
ninja -C .x86 clean
```

- Start the FW in x86:

- \${HSC\_DIR}/spica5/firmware/application/.x86/spica5\_fw\_app --ip localhost --port 50000 --cli-port 52000

# Compile the API and CUNIT test with debug flag

- Clean CAPI and test:

- [dle@dc5lp-vinetx076 capi]\$ make -C \${HSC\_DIR}/spica5/api/capi clean
- [dle@dc5lp-vinetx076 capi]\$ make -C \${HSC\_DIR}/spica5/api/capi/tests clean

- Make universal api clib:

- [dle@dc5lp-vinetx076 capi]\$ make -C \${HSC\_DIR}/spica5/api/capi PRODUCTS=spica5 PROJECT\_PREFIX=spica5 CC\_OPTIONS="-g3 -O0" legacy -j5

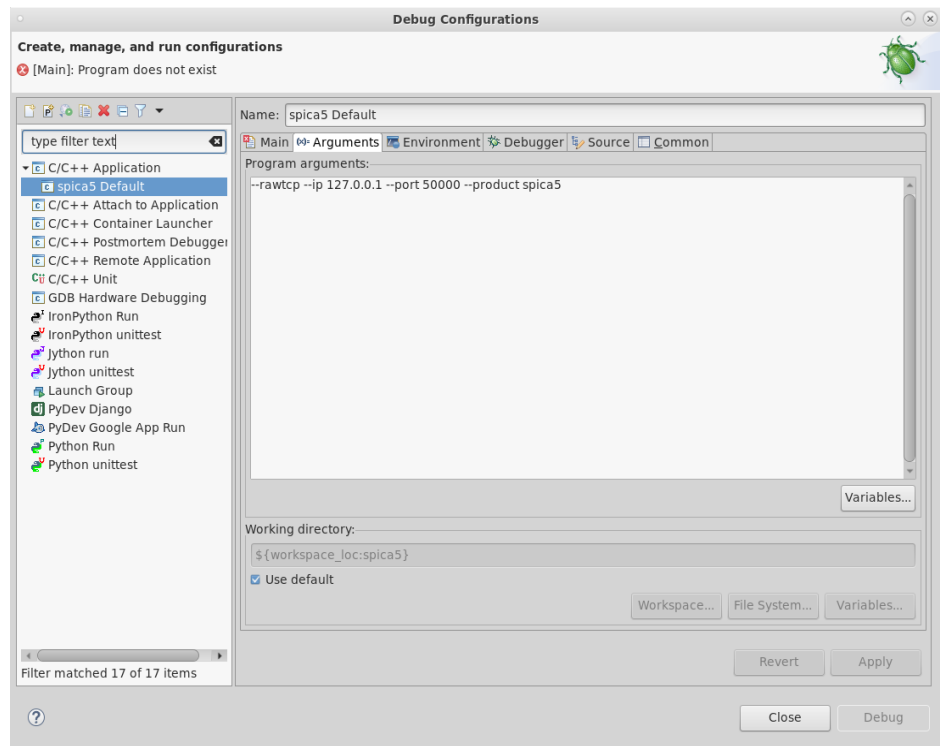
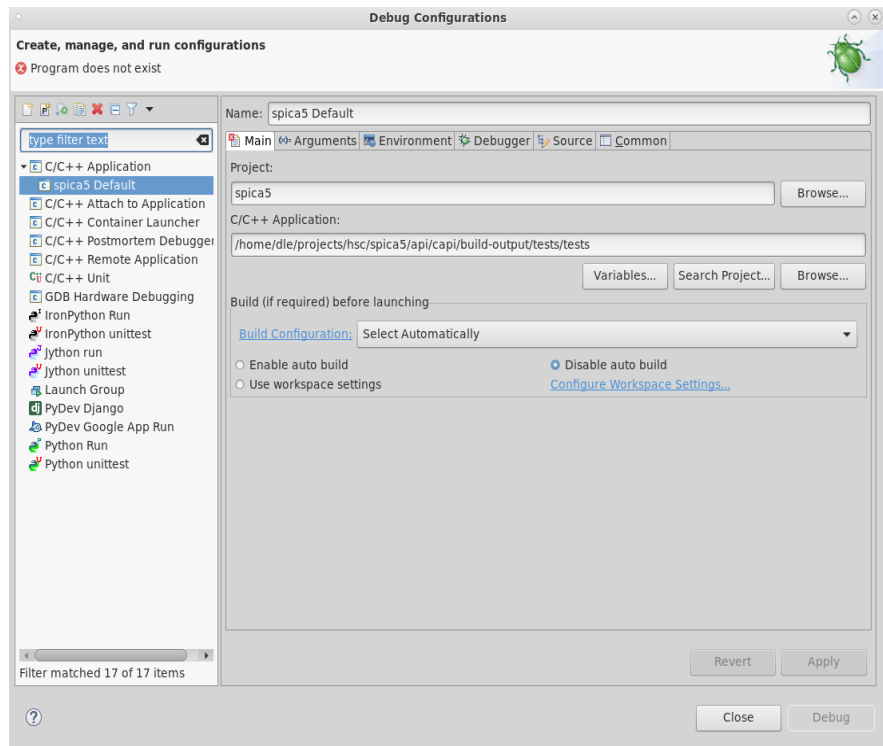
- Make legacy api clib:

- [dle@dc5lp-vinetx076 capi]\$ make -C \${HSC\_DIR}/spica5/api/capi PRODUCTS=spica5\_spicap PROJECT\_PREFIX=spica5 CC\_OPTIONS="-g3 -O0" legacy -j5

- Make the unit test:

- [dle@dc5lp-vinetx076 capi]\$ make -C \${HSC\_DIR}/spica5/api/capi/tests xmlrpc
- [dle@dc5lp-vinetx076 capi]\$ make -C \${HSC\_DIR}/spica5/api/capi/tests PROJECT\_PREFIX=spica5 CC\_OPTIONS="-g3 -O0" DEBUG=1 -j 3

# Debug the api, test program with the gdb



# Debug the api, test program with the gdb

The screenshot shows a GDB debugging session in an IDE. The left pane displays the Project Explorer with a list of threads and stack frames. The main pane shows the source code of `spica5_register.c`, with a yellow tooltip displaying the value of the `*addr` variable. The tooltip shows the variable's name, details, default, decimal, hex, binary, and octal values. The bottom pane shows a console output with a table of test channels.

**Project Explorer:**

- spica5 Default [C/C++ Application]
- tcsh [20817] [cores: 1]
- Thread #1 [tests] 20817 [core: 1] (Suspended : Step)
- spica\_rebase\_by\_addr() at spica5\_reg\_access.c:206 0x40ef9d
- spica\_reg\_channel\_read() at spica5\_reg\_access.c:265 0x40f036
- is\_squelched() at spica5\_chn.c:259 0x411387
- hsc\_chn\_is\_squelched() at hsc\_chn.c:394 0x407930
- test\_channel\_squelch\_config() at spica5\_tcasc\_chn.c:53 0x40485a
- test\_hsc\_channel\_squelch\_config() at spica5\_tcasc\_chn.c:62 0x4049d4
- tcasc\_run\_tfun\_nofork.isra.11() at 0x7ffff7990d12
- srunner\_run() at 0x7ffff7990f6e
- test\_channels() at spica5\_tests.c:93 0x4021c1
- tests() at spica5\_tests.c:101 0x402237
- <...more frames...>
- gdb (7.6.1)

**Source Code (spica5\_register.c):**

```
184 *
185 */
186= inphi_status_t spica_rebase_by_addr(
187 uint32_t* die,
188 uint32_t* channel,
189 uint32_t* addr)
190 {
191 inphi_status_t status = INPHI_OK;
192
193 // DEBUG BRAD: For now don't support two EFUSE maps since they cause
194 // a recursive loop
195 if(*addr >= 0x80400 && *addr <= 0x80a00)
196 {
197 return status;
198 }
199
200 const spica_channel_info_t* info = spica_channel_info(*addr);
201
202 *addr = *addr + (*channel * info->span);
203
204 // *addr
205 // *addr
206
207
208 }
209
210= /**
211 * Th
212 * on
213 *
214 * @p
215 * @p
216 * @p
```

**Variable Value:**

| Expression | Type      | Value         |
|------------|-----------|---------------|
| *addr      | uint32_t* | 0x7ffffff8254 |
| *addr      | uint32_t  | 1365067       |

**Variable Details:**

Name : \*addr  
Details: 1365067  
Default: 1365067  
Decimal: 1365067  
Hex: 0x14d44b  
Binary: 101001101010001001011  
Octal: 05152113

**Console Output:**

| Test channels | no | no | no | 0 |
|---------------|----|----|----|---|
| 1             | no | no | no | 0 |
| 2             | no | no | no | 0 |
| 3             | no | no | no | 0 |
| 4             | no | no | no | 0 |
| 5             | no | no | no | 0 |
| 6             | no | no | no | 0 |
| 7             | no | no | no | 0 |



# Double check register with the ral\_client tool

The screenshot shows the Eclipse IDE with the `spica5_registers.h` file open. The file contains definitions for various registers, including `SPICA_POR_TX_TXD_ERRINJ_CFG1` and `SPICA_POR_TX_TXD_SQUELCH_EN_CFG`. The `SPICA_POR_TX_TXD_SQUELCH_EN_CFG` register is highlighted, showing its address `0x14d44b`.

A terminal window is open, showing the output of the `ral_client` tool. The command `ral_client --port 50000 rd 14d44b --decode` is executed, and the output shows the register's fields and values.

The terminal output is as follows:

```
--die INTEGER Die ID
--decode Display fields
--chip TEXT Chip type required only with --decode
--field REGISTER_FIELD Bit-field
--help Show this message and exit.

[venv] [d]@dc5lp-vineth076 scripts]$ ral_client --port 50000 rd 14d44b --decode
2022-05-16 21:08:10 dc5lp-vineth076.inphi-corp.local root[25937] ERROR Chip type must be specified t
g decode registers
[venv] [d]@dc5lp-vineth076 scripts]$ ral_client --port 50000 rd 14d44b --decode --chip spica5
2022-05-16 21:08:34 dc5lp-vineth076.inphi-corp.local root[1011] INFO Loading spica5 bitfields ...

address	name/fields
0x14d44b | REGISTER POR_TX_TXD_SQUELCH_EN_CFG t
-----|-----
addr: 0x14d44b (Dotted Decimal: 20.54347)
inst: 0 (total=9, span=0x1000)
val: 0x1
die: 0x0 (lower 8 bits = instance, upper 24 bits = ASIC)
fields:
-----|-----
RSRV01[15:0]: 0x0
SQUELCH_EN[00:00]: 0x1

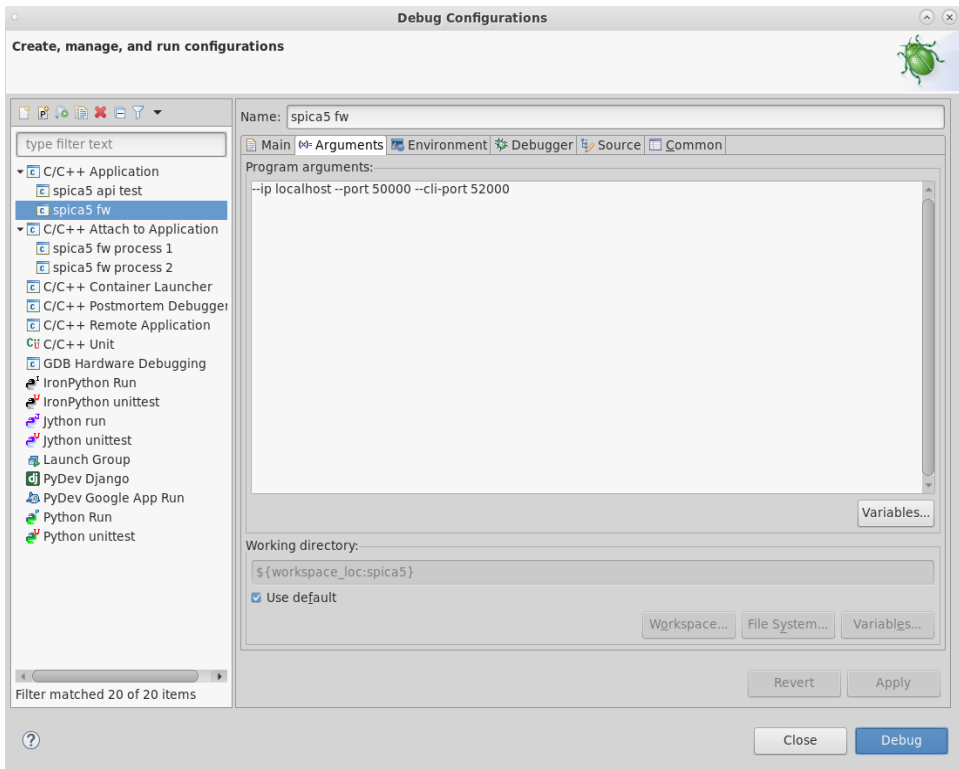
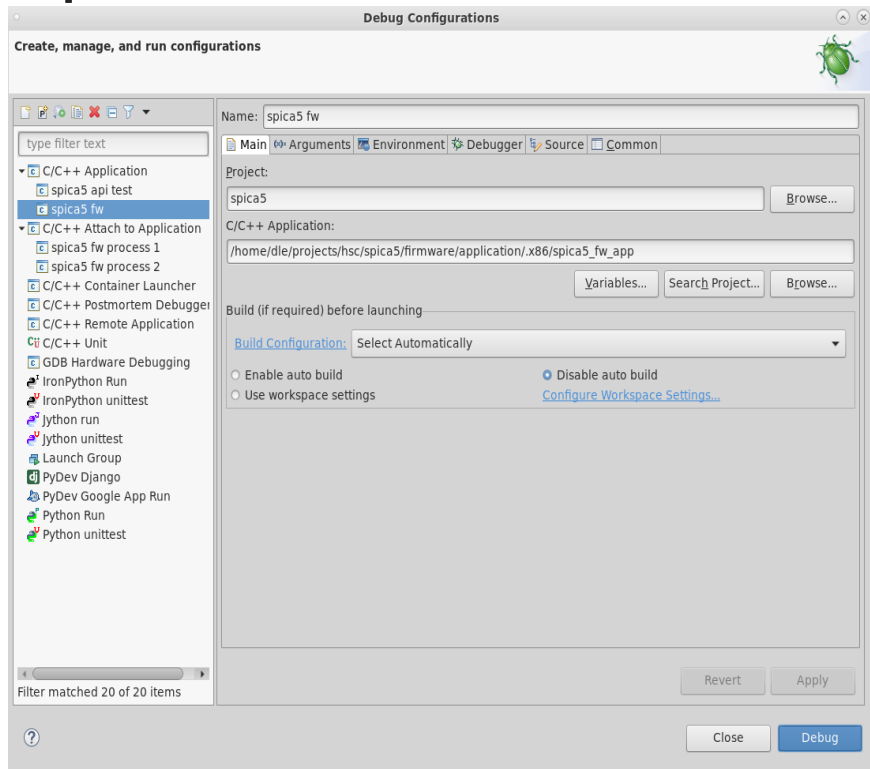
[venv] [d]@dc5lp-vineth076 scripts]$ pwd
/home/dle/projects/hsc/spica5/tools/pyral/scripts
[venv] [d]@dc5lp-vineth076 scripts]$
```

The IDE also shows a console window with the following output:

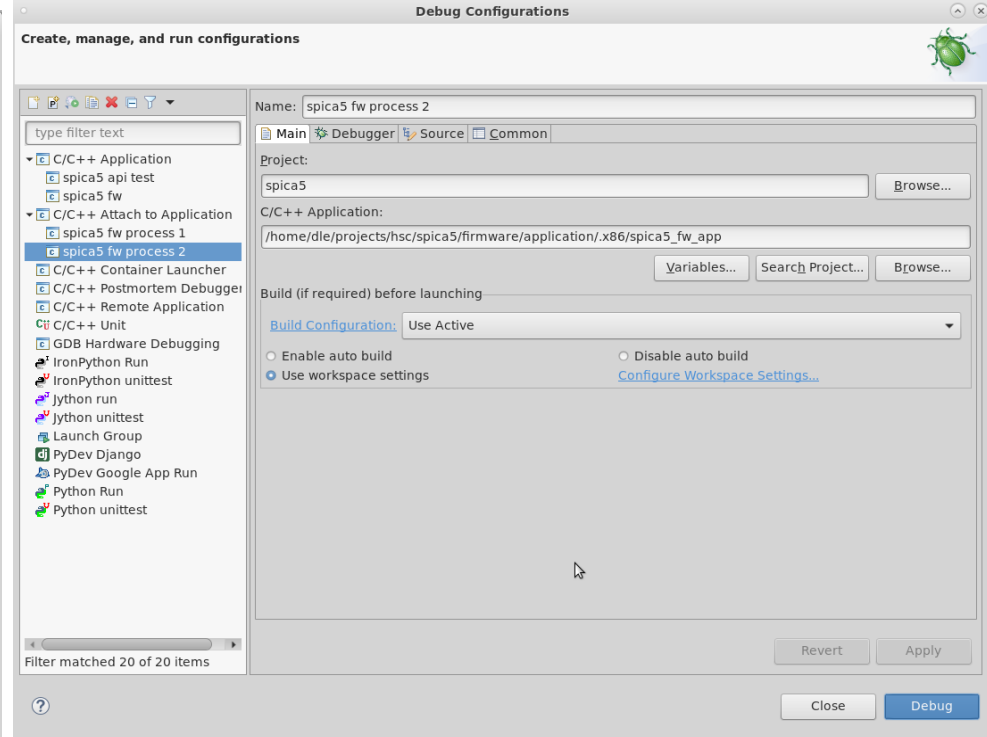
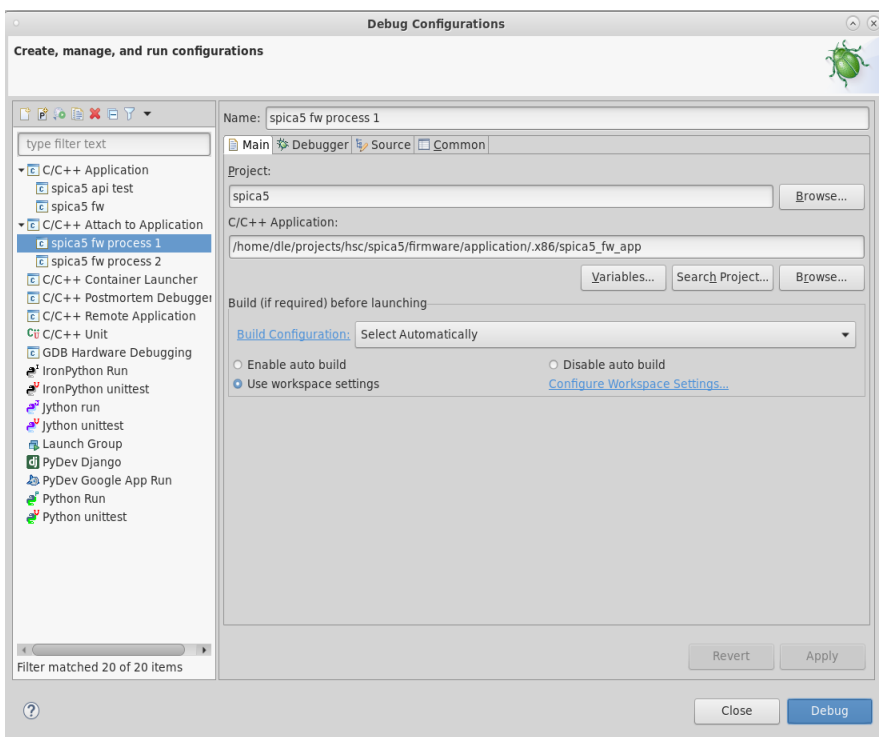
```
1 | no | no | no | 0
2 | no | no | no | 0
3 | no | no | no | 0
4 | no | no | no | 0
5 | no | no | no | 0
6 | no | no | no | 0
7 | no | no | no | 0

Test channels
```

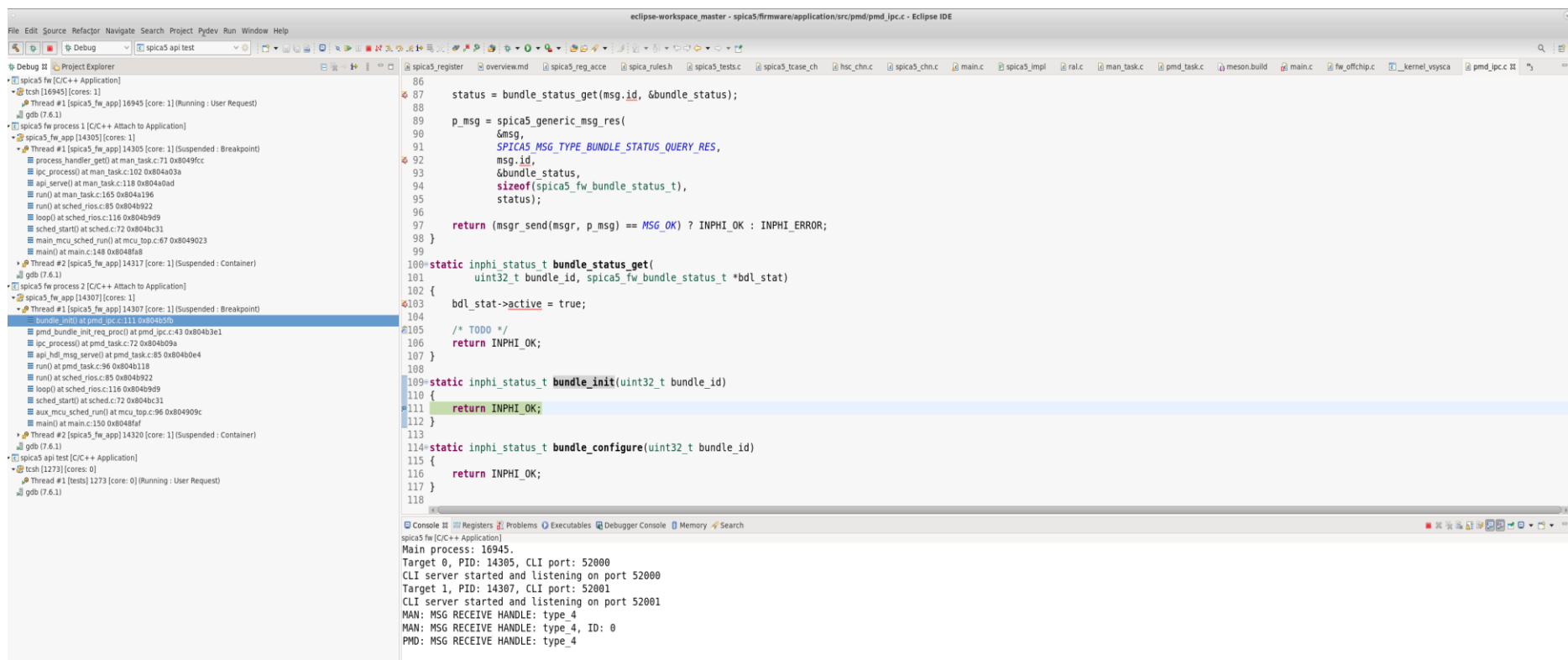
# Debug the FW with gdb: setup eclipse for the main process



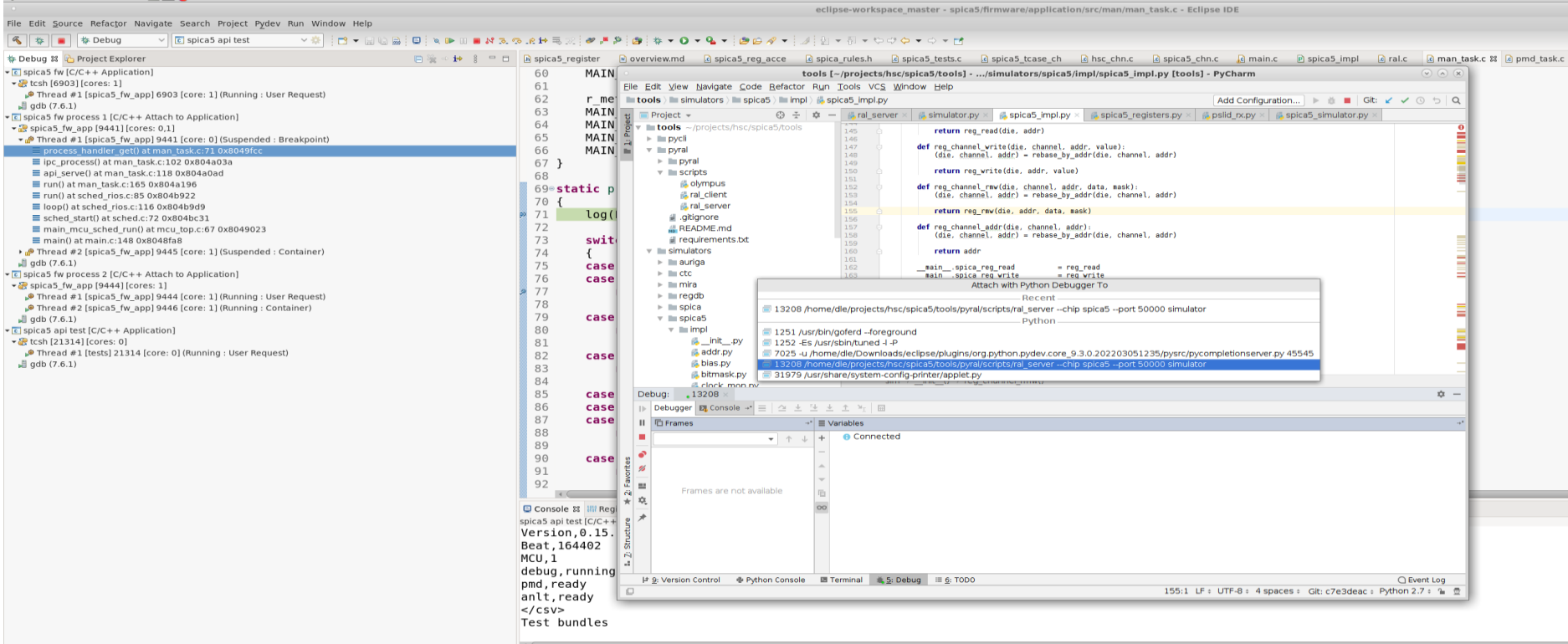
# Debug the FW with gdb: setup eclipse for the 2 forked processes (pmd\_task, man\_task)



# Debug the FW with gdb: api unit test <-> fw man task process <-> pmd task process

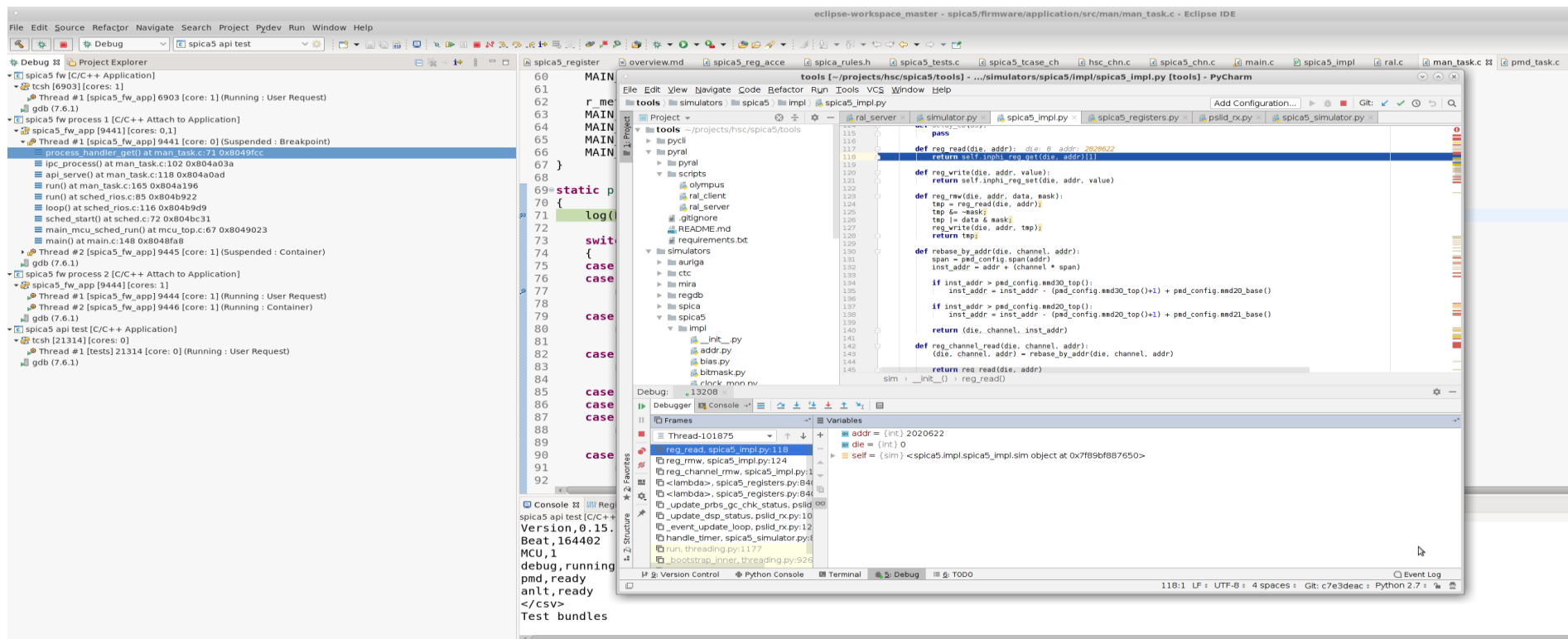


# Debug the python simulator with pycharm: run ral\_server with the simulator and attach the python process to the pycharm (from the run menu)



# Debug the python simulator with pycharm:

api unit test <-> fw man task process <-> pmd task process  
<-> simulator



# Tools in dc5lp

- Pycharm:

- `/tools/cad/jetbrains/pycharm/2019.13/Linux/bin/pycharm.sh`

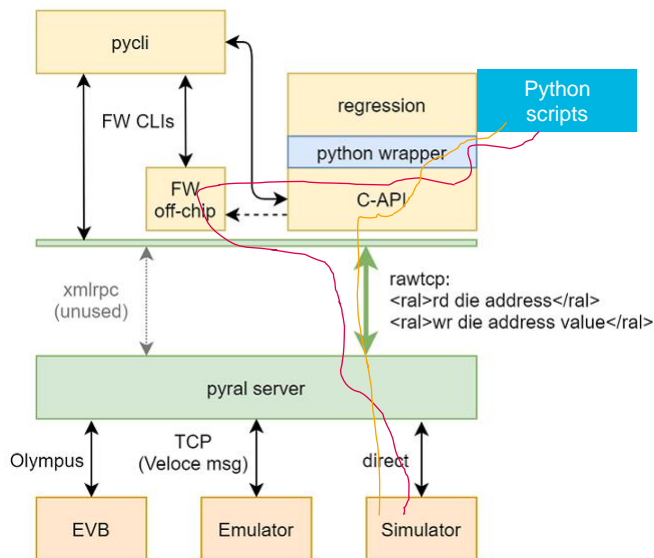
- Python3:

- `/tools/cad/python3.9.12/bin/python3`
- `setenv PATH /tools/cad/python3.9.12/bin/:$PATH`

- Create python3 virtual environment (so that you can use pip to install any python packages):

- `[dle@dc5lp-vinetx076]$ python -m venv python3.9_venvok`
- `source python3.9_venvok/bin/activate.csh`
- `pip install -r ${HSC_DIR}/spica5/tools/pyral/requirements.txt`

# Use python3 with swig wrapper python3 lib to generate Python API and run python3 api sample script with 2 mcus



## ▪ Start pyral along with simulator:

- `source ~/python3.9_venvok/bin/activate.csh`
- `python ${HSC_DIR}/spica5/tools/pyral/scripts/ral_server --chip spica5 --port 50000 simulator`

## ▪ Start off-chip FW:

- `[dle@dc5lp-vinetx028 ~]$ ${HSC_DIR}/spica5/firmware/application/x86/spica5_fw_app --ip localhost --port 50000 --cli-port 52000`

## ▪ 3 options to make api to generate python lib

### - Generate both universal api and legacy api:

- `[dle@dc5lp-vinetx028 capi]$ make -C ${HSC_DIR}/spica5/api/capi PRODUCTS=spica5 PROJECT_PREFIX=spica5 pyversion=3.9 python`

### - Generate python lib for universal api only:

- `[dle@dc5lp-vinetx028 capi]$ make -C ${HSC_DIR}/spica5/api/capi PRODUCTS=spica5 PROJECT_PREFIX=spica5 pyversion=3.9 pythonuapi`

### - Generate python lib for legacy api only:

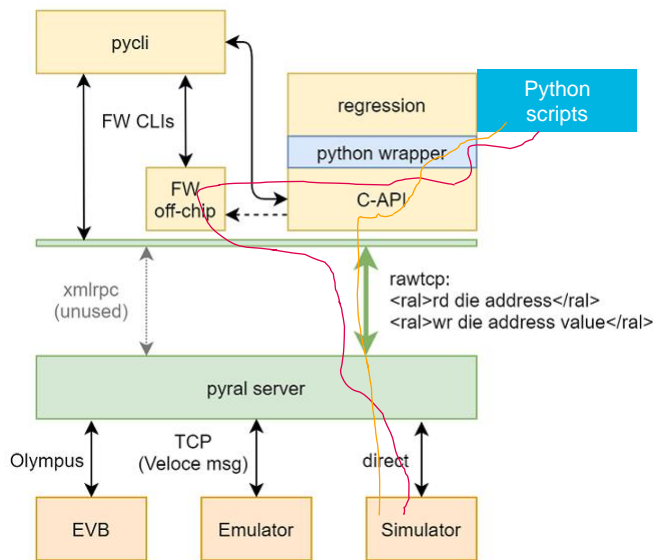
- `[dle@dc5lp-vinetx028 capi]$ make -C ${HSC_DIR}/spica5/api/capi PRODUCTS=spica5 PROJECT_PREFIX=spica5 pyversion=3.9 pythoncapi`

## ▪ Go to examples/py/spica5 in the python3 venv to run an example python script:

- `cd ${HSC_DIR}/spica5/api/capi/examples/py/spica5`
- `setenv PYTHONPATH /home/dle/projects/hsc/spica5/tools/pyral/pyral:${PYTHONPATH}`
- Run universal api example: `python odsp_opaq_api_sample.py`
- Run legacy api example: `python odsp_api_sample.py`



# Use python3 with swig wrapper python3 lib to generate Python API and run python3 api sample script with 1 cmu framework



## ▪ Start pyral along with simulator:

- `source ~/python3.9_venvok/bin/activate.csh`
- `python ${HSC_DIR}/spica5/tools/pyral/scripts/ral_server --chip spica5 --port 50000 simulator`

## ▪ Start off-chip FW:

- `[dle@dc5lp-vinetx028 ~]$ ${HSC_DIR}/spica5/firmware/application/x86/spica5_fw_app --ip localhost --port 50000 --cli-port 52000 --mcu-id 0`

## ▪ 3 options to make api to generate python lib

### - Generate both universal api and legacy api:

- `[dle@dc5lp-vinetx028 capi]$ make -C ${HSC_DIR}/spica5/api/capi PRODUCTS=spica5 PROJECT_PREFIX=spica5 pyversion=3.9 python`

### - Generate python lib for universal api only:

- `[dle@dc5lp-vinetx028 capi]$ make -C ${HSC_DIR}/spica5/api/capi PRODUCTS=spica5 PROJECT_PREFIX=spica5 pyversion=3.9 pythonuapi`

### - Generate python lib for legacy api only:

- `[dle@dc5lp-vinetx028 capi]$ make -C ${HSC_DIR}/spica5/api/capi PRODUCTS=spica5 PROJECT_PREFIX=spica5 pyversion=3.9 pythoncapi`

## ▪ Go to examples/py/spica5 in the python3 venv to run an example python script:

- `cd ${HSC_DIR}/spica5/api/capi/examples/py/spica5`
- `setenv PYTHONPATH /home/dle/projects/hsc/spica5/tools/pyral/pyral:${PYTHONPATH}`
- Run universal api example: `python odsp_opaq_api_sample_1mcu.py`
- Run legacy api example: `python odsp_api_sample.py`

Ref: overview.md



Essential technology, done right™



MARVELL™