Laboratório Nacional de Luz Síncrotron

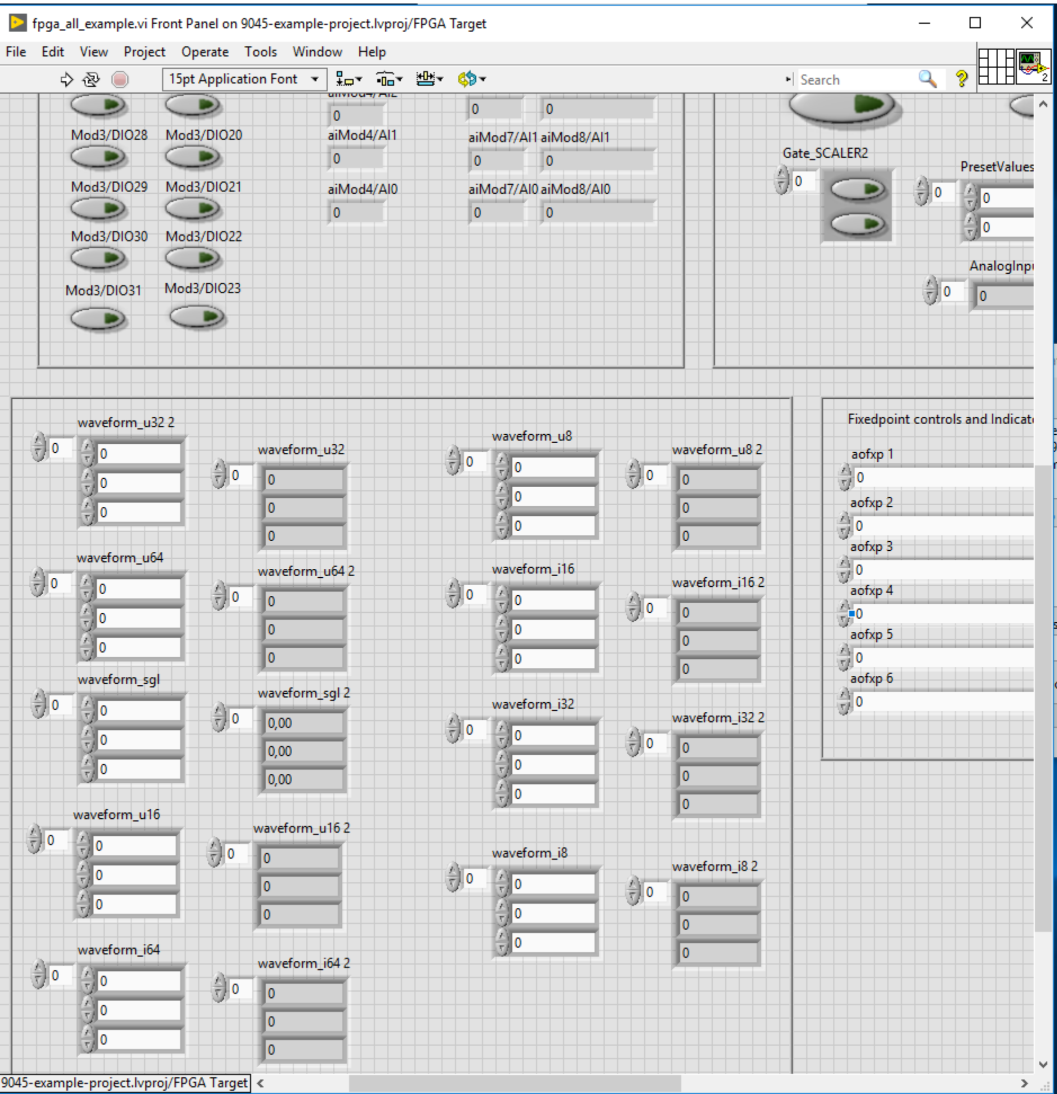
Project Nheengatu: EPICS support for CompactRIO FPGA and LabVIEW-RT

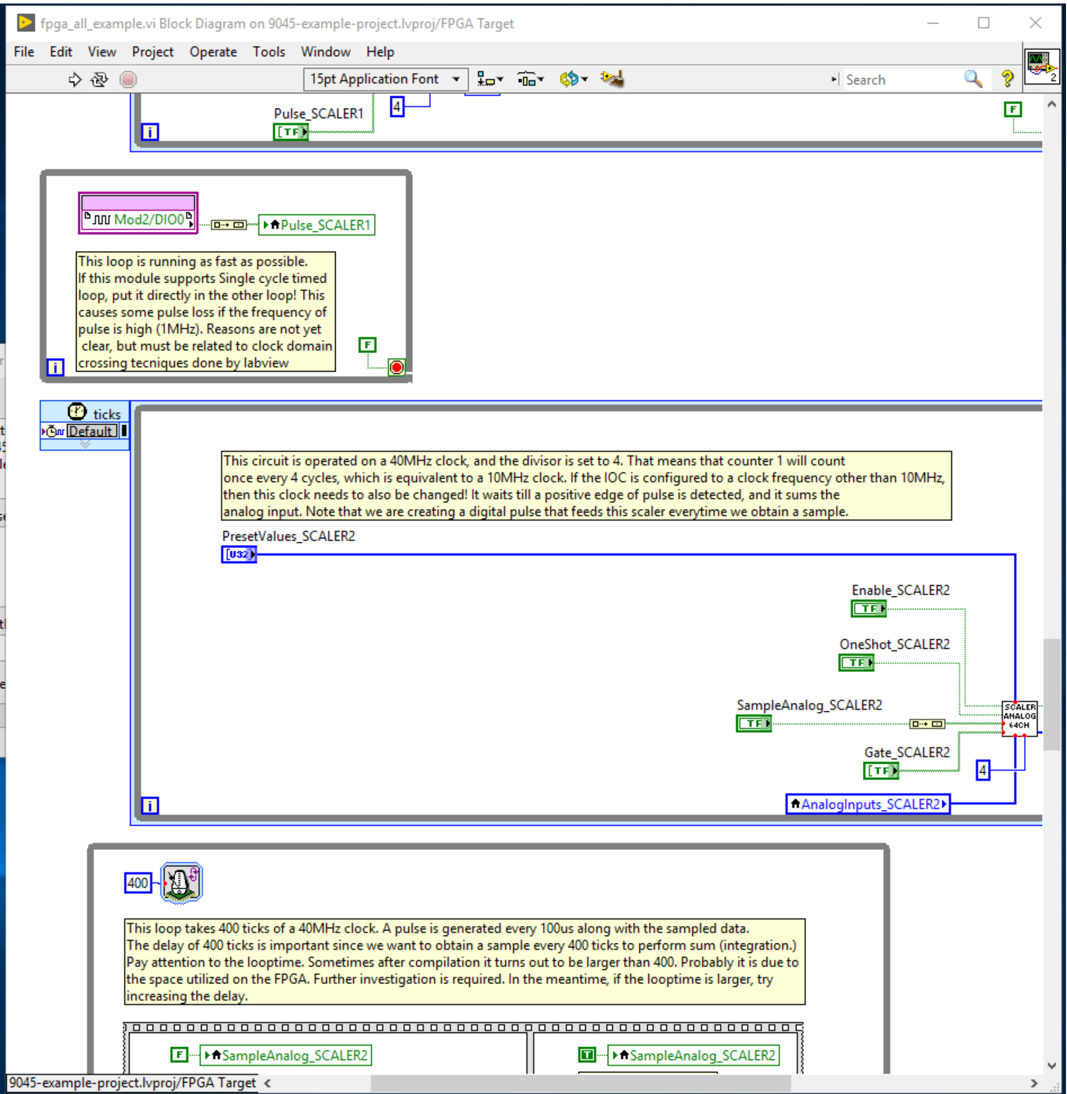
Dawood Alnajjar - SOL

Laboratório Nacional de Luz Síncrotron

LabVIEW

National instruments general interface to configure the CRIO through VI development. VIs Can be implmented on the FPGA or LinuxRT (labviewRT).





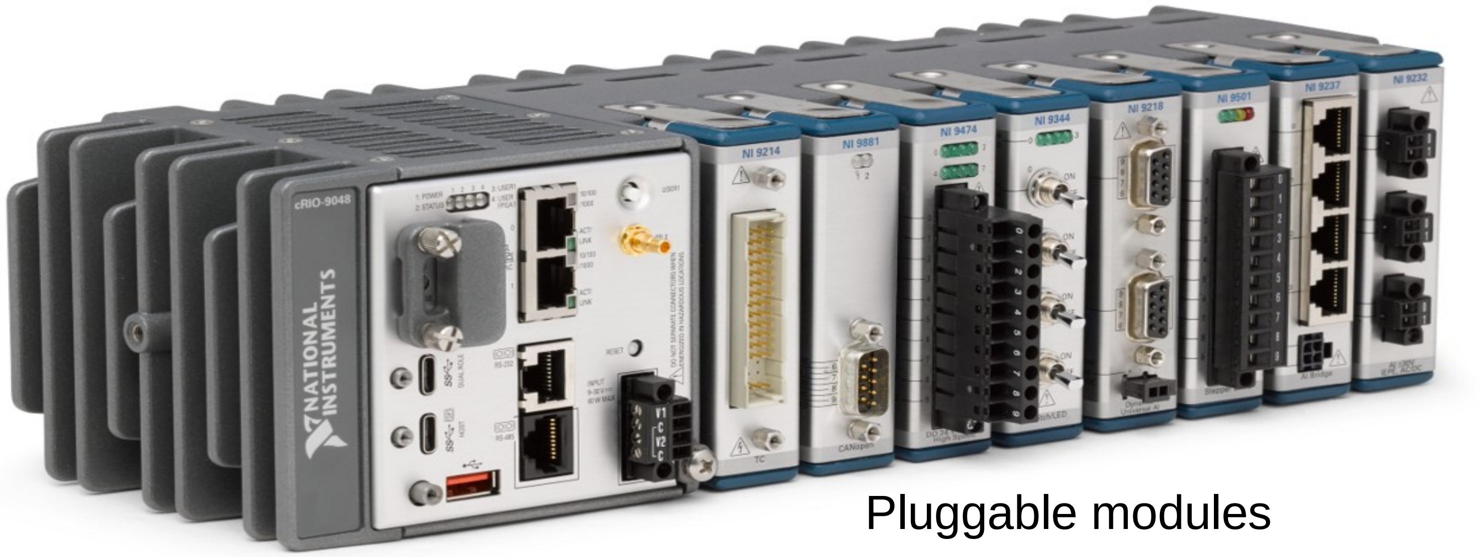
This loop is running as fast as possible. If this module supports Single cycle timed loop, put it directly in the other loop! This causes some pulse loss if the frequency of pulse is high (1MHz). Reasons are not yet clear, but must be related to clock domain crossing tecniques done by labview

This circuit is operated on a 40MHz clock, and the divisor is set to 4. That means that counter 1 will count once every 4 cycles, which is equivalent to a 10MHz clock. If the IOC is configured to a clock frequency other than 10MHz, then this clock needs to also be changed! It waits till a positive edge of pulse is detected, and it sums the analog input. Note that we are creating a digital pulse that feeds this scaler everytime we obtain a sample.

This loop takes 400 ticks of a 40MHz clock. A pulse is generated every 100us along with the sampled data. The delay of 400 ticks is important since we want to obtain a sample every 400 ticks to perform sum (integration.) Pay attention to the looptime. Sometimes after compilation it turns out to be larger than 400. Probably it is due to the space utilized on the FPGA. Further investigation is required. In the meantime, if the looptime is larger, try increasing the delay.

CompactRIO 9045

Has a Xilinx Kintex 7 FPGA and an atom processor running LinuxRT



Pluggable modules

Objective

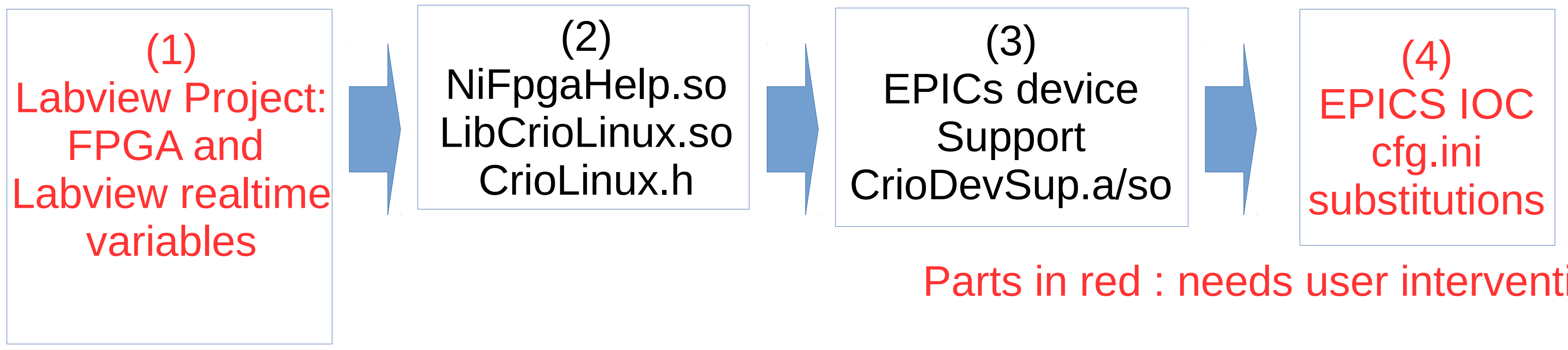
Input and output binary and analog data, fast trigger counting,
fine control using EPICS and CRIO

FPGA
LabviewRT (runs in LinuxRT)

Users

EPICS

Nheengatu architecture



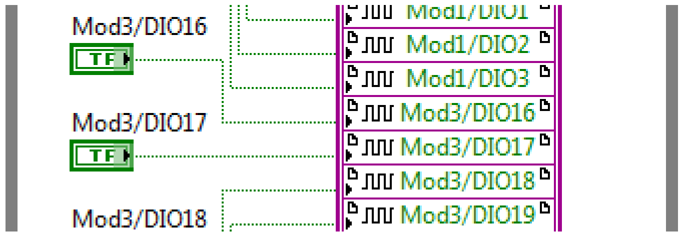
Parts in red : needs user intervention

(1) LabviewRT : Developed extensions to open a shared memory and write/read from/to it
FPGA : Use C API generator to generate addresses associated with variables
(2) LibCrioLinux.so : abstraction layer to reads and writes to shared memory and FPGA addresses of the variables using a reference “name”
NiFpgaHelp.so : Contains all NI specific functions
(3) Device support : EPICS library that contains 5 types of records : BI, BO, AI, AO, Scaler, Waveform, and contains functions to initialize CRIO from IOC
(4) EPICS IOC : IOC (no custom code, just db templates) that has access to all device support types

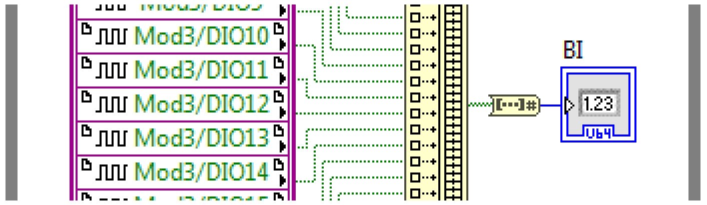
1.1 Handling FPGA variables (1/6)

- **BIs: Concatenate and connect to U64 Indicator**
- **BOs: As is (boolean indicator)**

Binary output : as is



Binary input : Convert to 64-bit U64

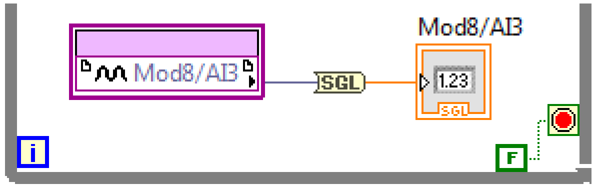


1.1 Handling FPGA variables (2/6)

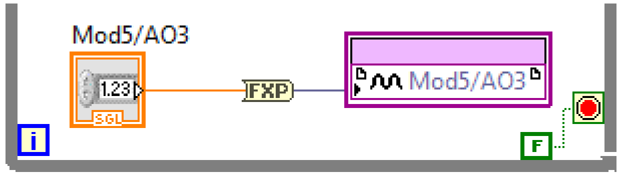
AI and AO handling : Convert to fixed-point or single precision floating point

Single precision floating point handling

Analog input : Convert to float

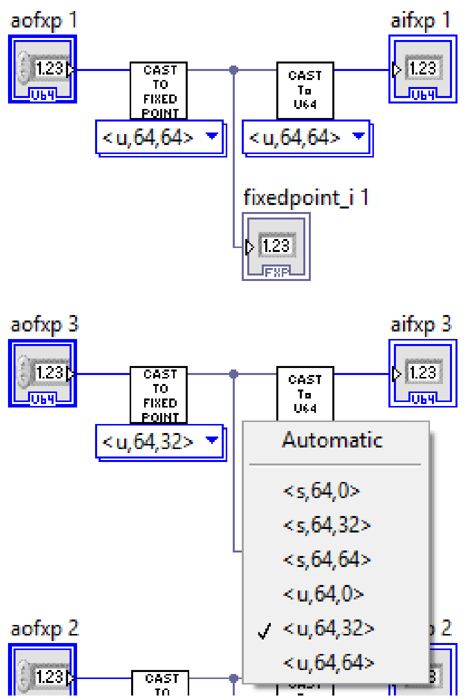


Analog output : Convert to FXP



Fixed-point handling

Analog input : use cast to U64 polymorphic VI



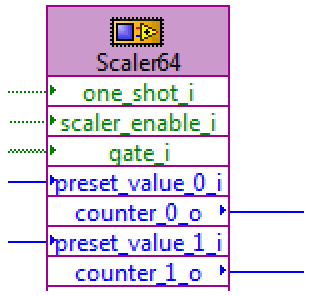
Analog output : use cast to fixed-point polymorphic VI

Laboratório Nacional de Luz Síncrotron

1.1 FPGA variables (3/6)

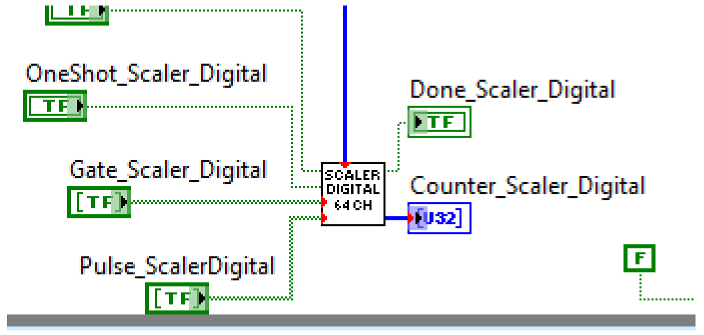
Scaler

Scaler IP in VHDL

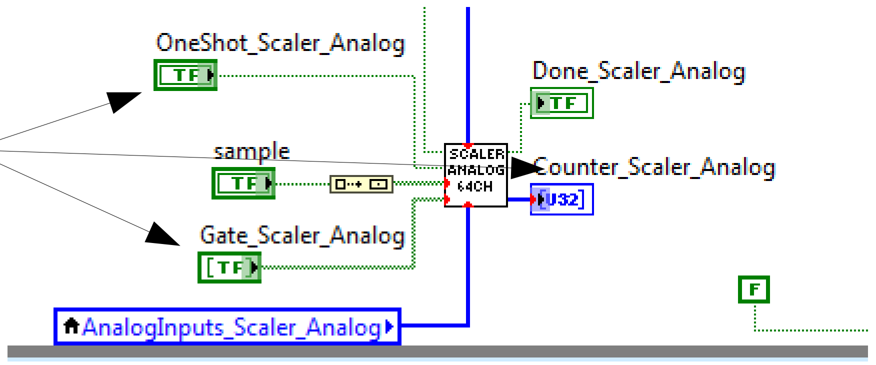


Validated with testbench

Scaler digital



Scaler Analog



If not all 64 counters are used, the logic can simply be optimized by reducing the size of the fixed input arrays.

FPGA utilization can be reduced by approximately 15% if this optimization is applied

1.1 Handling FPGA variables (4/6)

Reading arrays : instantiate indicator array

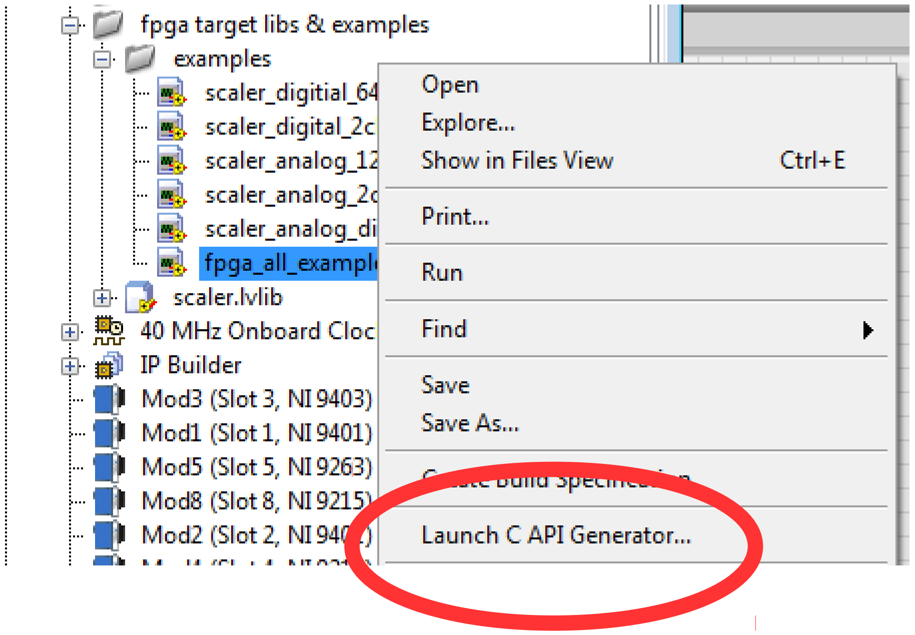
- **I08 I16, I32, I64**
- **U08, U16, U32, U64**
- **Single precision floating point**

Laboratório Nacional de Luz Síncrotron

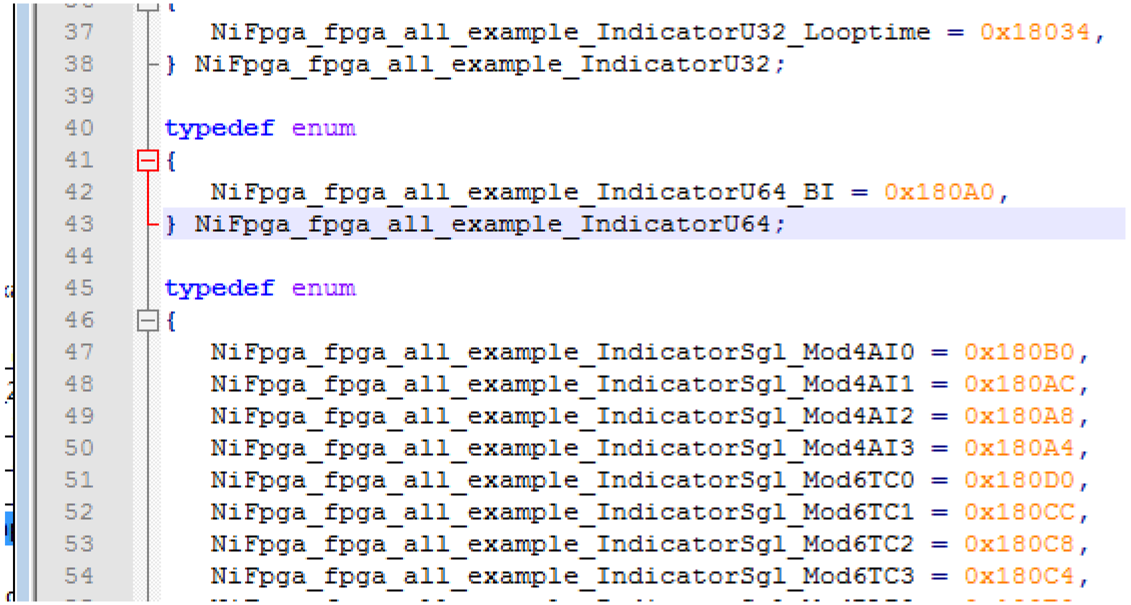
CNPEM

1.1 FPGA variables (5/6)

Generate bitstream and C API files



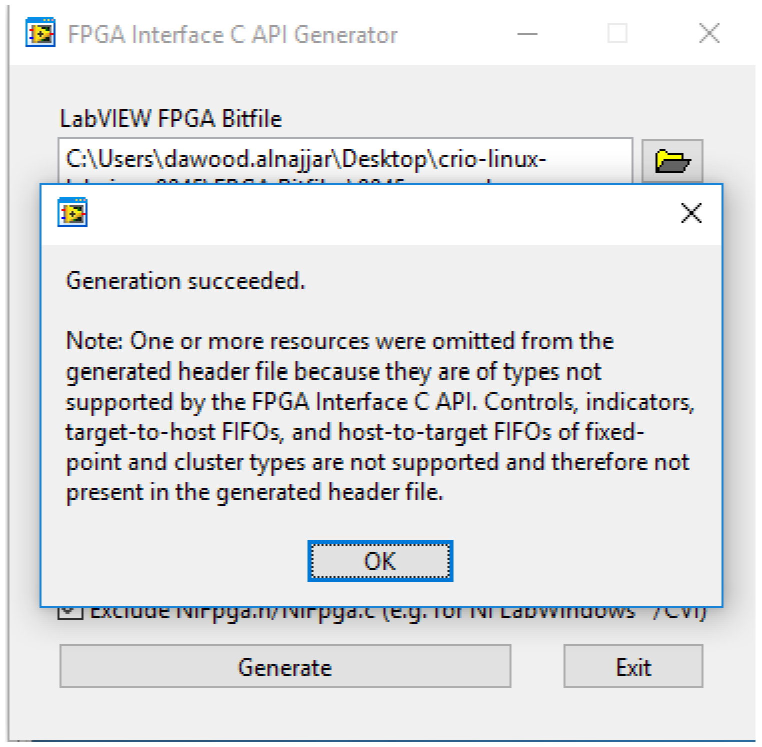
Generates header file with addresses



```
   NiFpga_fpga_all_example_IndicatorU32_Looptime = 0x18034,
} NiFpga_fpga_all_example_IndicatorU32;

typedef enum
{
   NiFpga_fpga_all_example_IndicatorU64_BI = 0x180A0,
} NiFpga_fpga_all_example_IndicatorU64;

typedef enum
{
   NiFpga_fpga_all_example_IndicatorSgl_Mod4AI0 = 0x180B0,
   NiFpga_fpga_all_example_IndicatorSgl_Mod4AI1 = 0x180AC,
   NiFpga_fpga_all_example_IndicatorSgl_Mod4AI2 = 0x180A8,
   NiFpga_fpga_all_example_IndicatorSgl_Mod4AI3 = 0x180A4,
   NiFpga_fpga_all_example_IndicatorSgl_Mod6TC0 = 0x180D0,
   NiFpga_fpga_all_example_IndicatorSgl_Mod6TC1 = 0x180CC,
   NiFpga_fpga_all_example_IndicatorSgl_Mod6TC2 = 0x180C8,
   NiFpga_fpga_all_example_IndicatorSgl_Mod6TC3 = 0x180C4,
```

The address of each variable must be noted for the cfg.ini file (or automation tools can be used).

1.1 FPGA variables (6/6)


FPGA Interface C API Generator

LabVIEW FPGA Bitfile

C:\Users\dawood.alnajjar\Desktop\crio-linux-

Generation succeeded.

Note: One or more resources were omitted from the generated header file because they are of types not supported by the FPGA Interface C API. Controls, indicators, target-to-host FIFOs, and host-to-target FIFOs of fixed-point and cluster types are not supported and therefore not present in the generated header file.

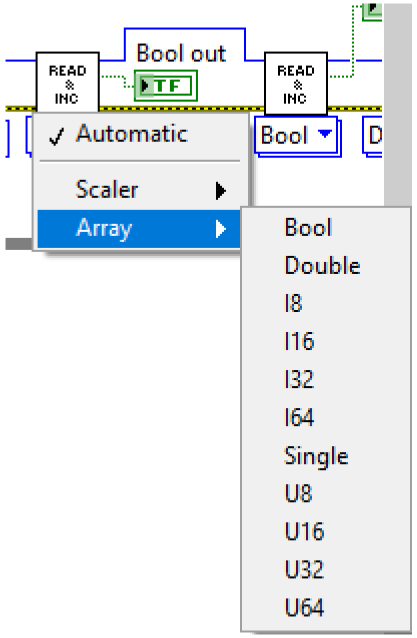
Make sure that non of your indicators and controls are omitted by abiding with the data types defined above!

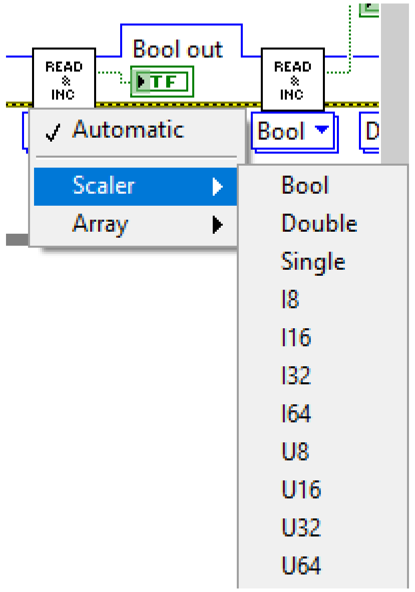
1.2 LabviewRT variables

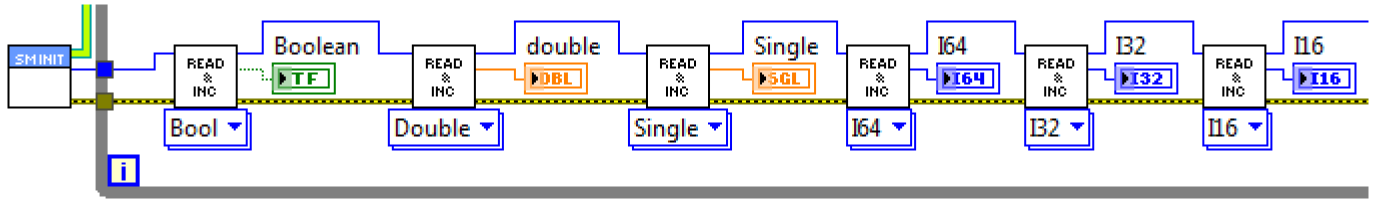
Four Vis were developed to synchronize variable/array exchange with EPICS

- IOC Shared memory initialize.vi
  - Initializes shared memory. All input can be left as default
- IOC Shared memory de-initialize.vi
  - de-initializes shared memory.
- SM read and increment.vi (polymorphic VI)
  - Reads the variable (**from EPICS**) when setting the polymorphic VI
- SM write and increment.vi (polymorphic VI)
  - Writes the variable (**to EPICS**) when setting the polymorphic VI

All VI MUST be chained with <ptr in> and <ptr out>.
The index of the variable must be noted for the cfg.ini file.





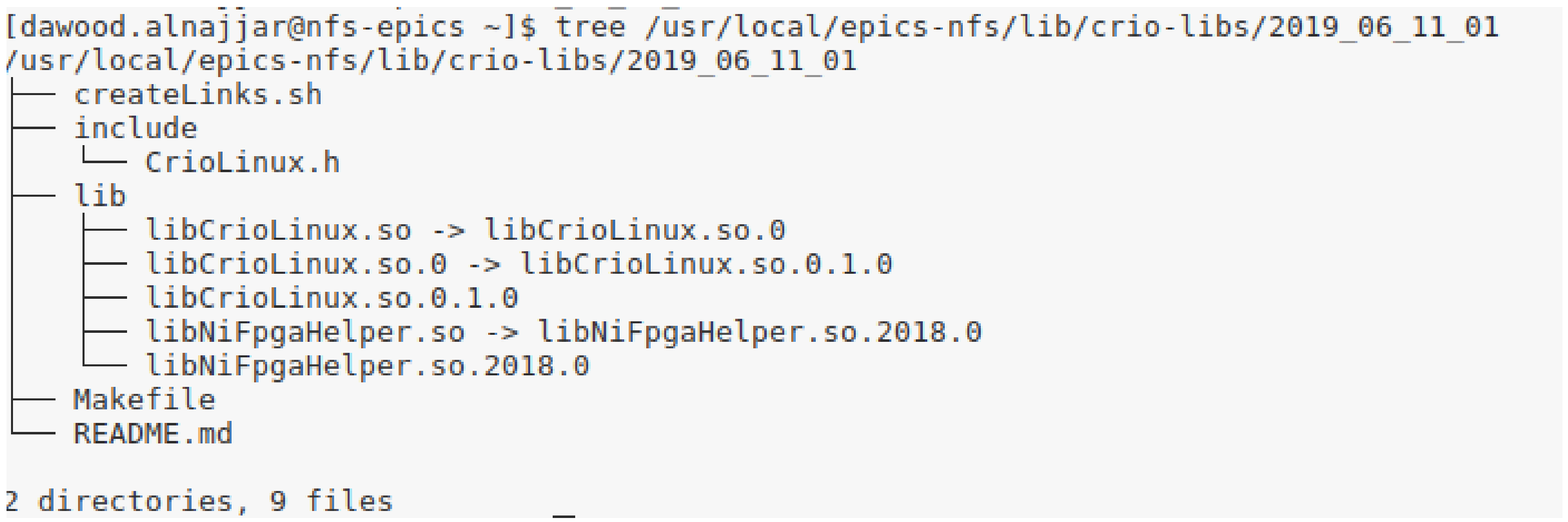


LNLS Laboratório Nacional de Luz Síncrotron sirius

CNPEM

2 & 3 libCrioLinux.so and EPICS device support

NO alteration or re-compile is required!



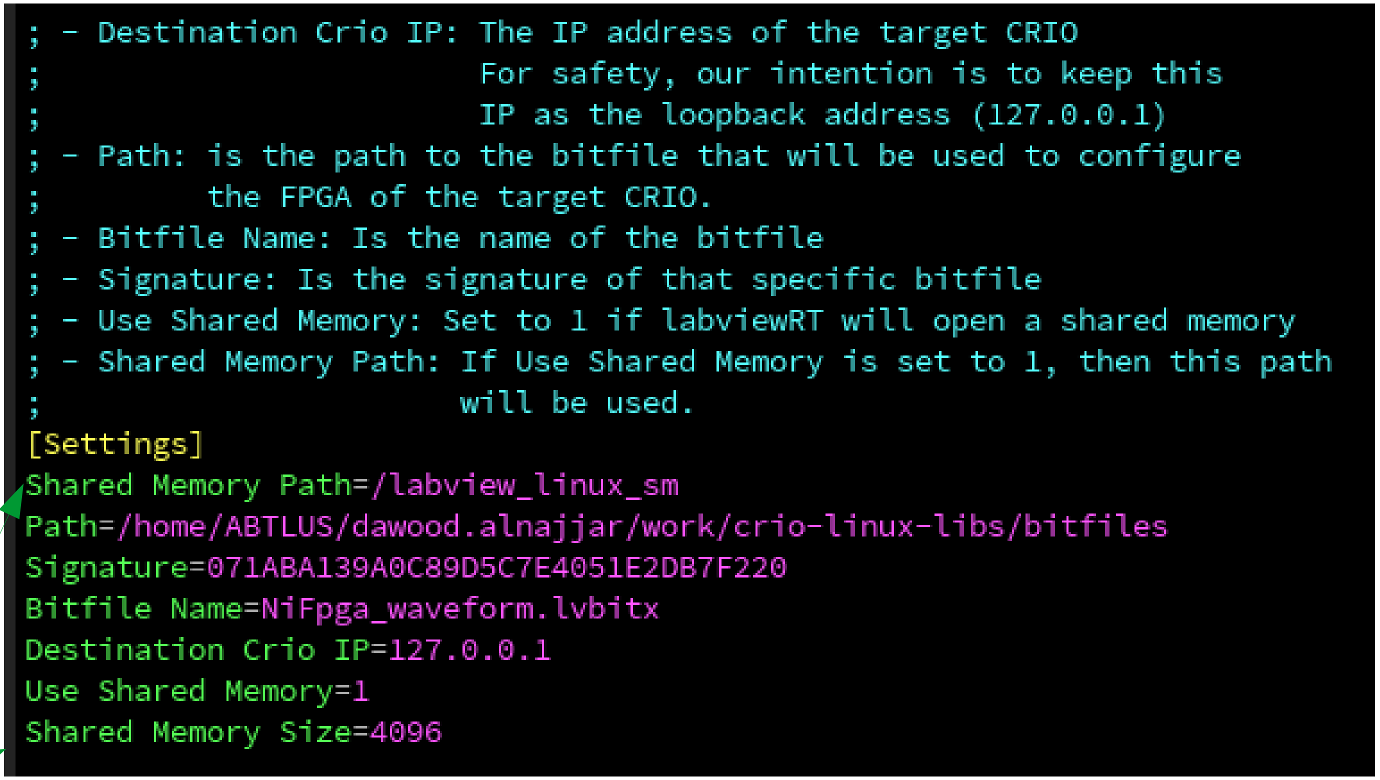
```
[dawood.alnajjar@nfs-epics ~]$ tree /usr/local/epics-nfs/lib/crio-libs/2019_06_11_01
/usr/local/epics-nfs/lib/crio-libs/2019_06_11_01
├── createLinks.sh
├── include
│   └── CrioLinux.h
├── lib
│   ├── libCrioLinux.so -> libCrioLinux.so.0
│   ├── libCrioLinux.so.0 -> libCrioLinux.so.0.1.0
│   ├── libCrioLinux.so.0.1.0
│   ├── libNiFpgaHelper.so -> libNiFpgaHelper.so.2018.0
│   └── libNiFpgaHelper.so.2018.0
├── Makefile
└── README.md

2 directories, 9 files
```

Device support library path needs to be indicated explicitely in the IOC

4.1 EPICS IOC (1/10)

As long as the associated DB files are used, the IOC does not need to be compiled either!

4.2 EPICS IOC – CFG.INI (2/10)



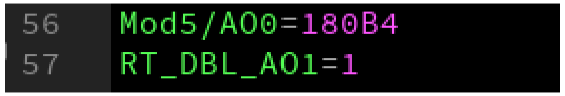
```
; - Destination Crio IP: The IP address of the target CRIO
;                        For safety, our intention is to keep this
;                        IP as the loopback address (127.0.0.1)
; - Path: is the path to the bitfile that will be used to configure
;         the FPGA of the target CRIO.
; - Bitfile Name: Is the name of the bitfile
; - Signature: Is the signature of that specific bitfile
; - Use Shared Memory: Set to 1 if labviewRT will open a shared memory
; - Shared Memory Path: If Use Shared Memory is set to 1, then this path
;                       will be used.
[Settings]
Shared Memory Path=/labview_linux_sm
Path=/home/ABTLUS/dawood.alnajjar/work/crio-linux-libs/bitfiles
Signature=071ABA139A0C89D5C7E4051E2DB7F220
Bitfile Name=NiFpga_waveform.lvbitx
Destination Crio IP=127.0.0.1
Use Shared Memory=1
Shared Memory Size=4096
```

If changed, must be changed in labview RT VI too!

Disabling shared memory disables all labview RT variable processing.

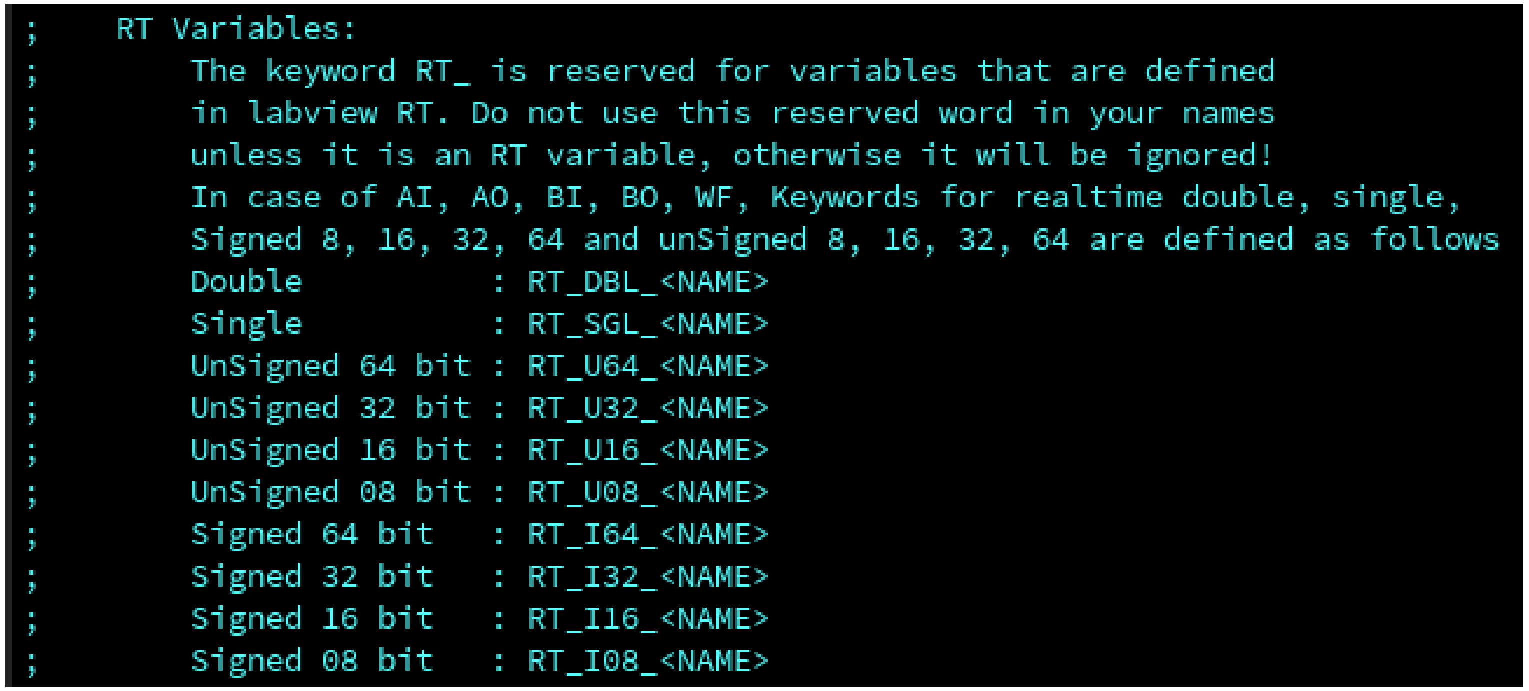
4.2 EPICS IOC – CFG.INI (3/10)

- 2 types of variables: labview RT variables, and FPGA variables



```
Mod5/AO0=180B4
RT_DBL_AO1=1
```

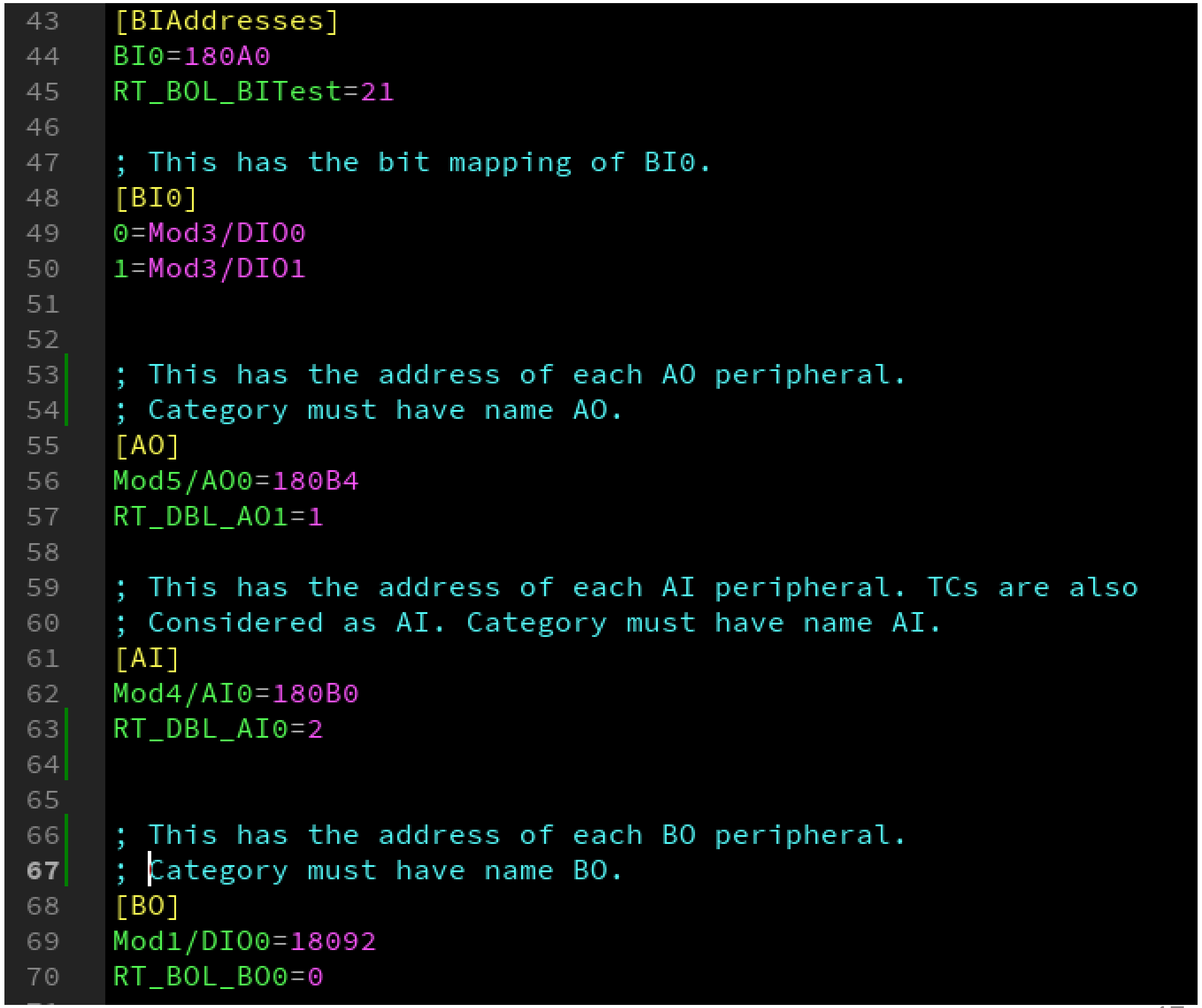
- The library distinguishes the RT variables type and its size through its name



```
;    RT Variables:
;        The keyword RT_ is reserved for variables that are defined
;        in labview RT. Do not use this reserved word in your names
;        unless it is an RT variable, otherwise it will be ignored!
;        In case of AI, AO, BI, BO, WF, Keywords for realtime double, single,
;        Signed 8, 16, 32, 64 and unSigned 8, 16, 32, 64 are defined as follows
;        Double          : RT_DBL_<NAME>
;        Single          : RT_SGL_<NAME>
;        UnSigned 64 bit : RT_U64_<NAME>
;        UnSigned 32 bit : RT_U32_<NAME>
;        UnSigned 16 bit : RT_U16_<NAME>
;        UnSigned 08 bit : RT_U08_<NAME>
;        Signed 64 bit   : RT_I64_<NAME>
;        Signed 32 bit   : RT_I32_<NAME>
;        Signed 16 bit   : RT_I16_<NAME>
;        Signed 08 bit   : RT_I08_<NAME>
```

4.2 EPICS IOC – CFG.INI (4/10)

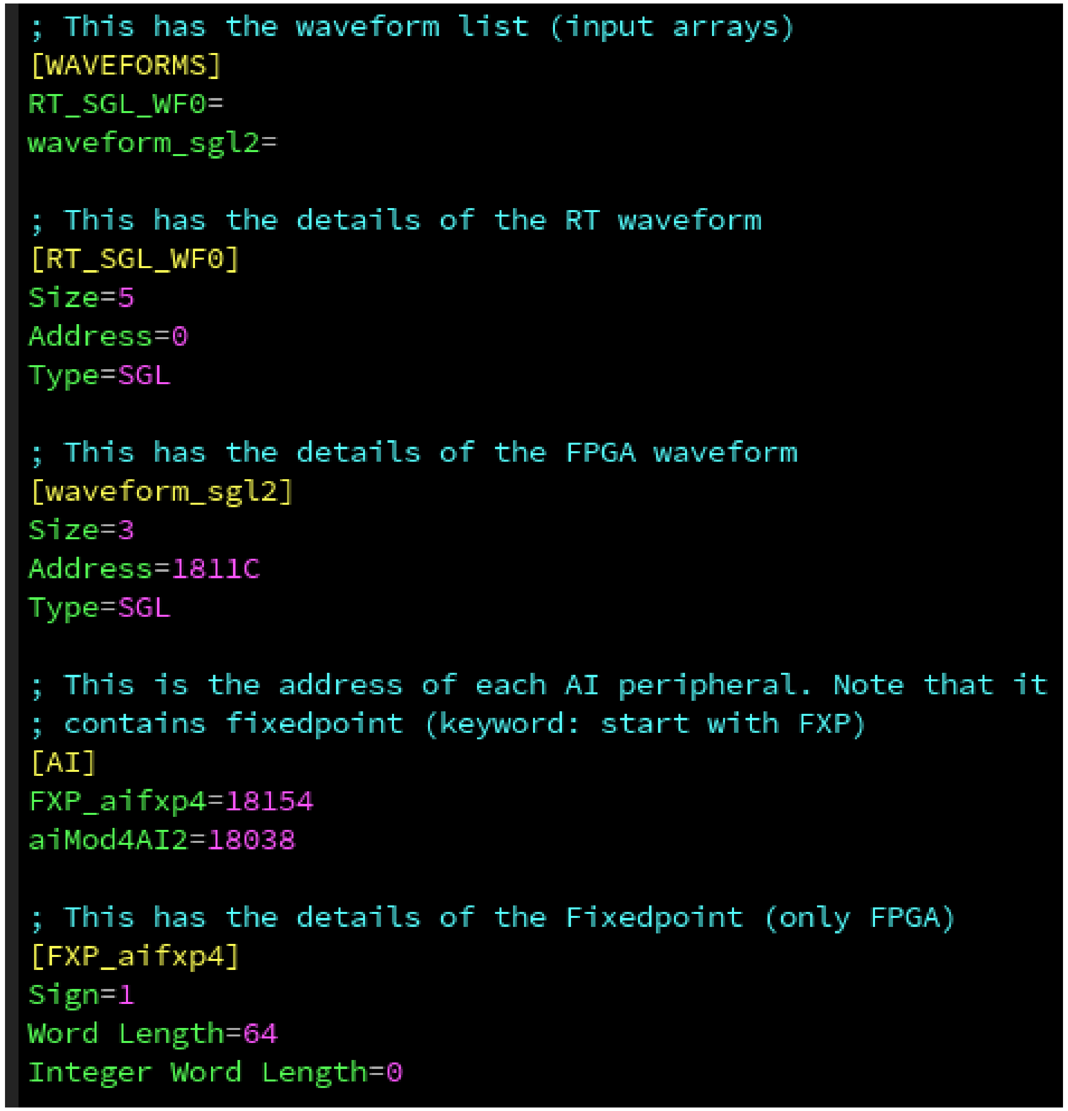
- [BIAddresses]
  - Address/index of BI
- [BI0]: index -name relation of each bit in the 64 bits (FPGA)
- [AO]: Address/index of each available output analog variable
- [AI]: Address/index of each available input analog variable
- [BO]: Address/index of each available output digital variable



```
[BIAddresses]
BI0=180A0
RT_BOL_BITest=21

; This has the bit mapping of BI0.
[BI0]
0=Mod3/DIO0
1=Mod3/DIO1


; This has the address of each AO peripheral.
; Category must have name AO.
[AO]
Mod5/AO0=180B4
RT_DBL_AO1=1

; This has the address of each AI peripheral. TCs are also
; Considered as AI. Category must have name AI.
[AI]
Mod4/AI0=180B0
RT_DBL_AI0=2


; This has the address of each BO peripheral.
; Category must have name BO.
[BO]
Mod1/DIO0=18092
RT_BOL_BO0=0
```

4.2 EPICS IOC – CFG.INI (5/10)

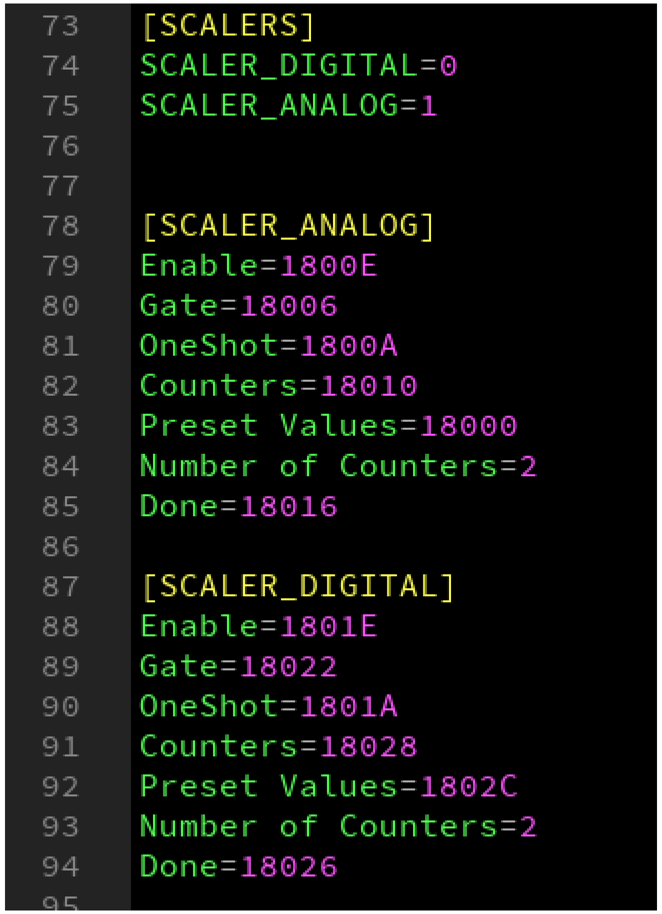
- [WAVEFORM]
  - List pf waveforms (RT & FPGA)
- [WAVEFORMXX]:
  - details of WAVEFORMXX
- [FXP_XX]:
  - Details of FXP_XX



```
; This has the waveform list (input arrays)
[WAVEFORMS]
RT_SGL_WF0=
waveform_sgl2=

; This has the details of the RT waveform
[RT_SGL_WF0]
Size=5
Address=0
Type=SGL

; This has the details of the FPGA waveform
[waveform_sgl2]
Size=3
Address=1811C
Type=SGL

; This is the address of each AI peripheral. Note that it
; contains fixedpoint (keyword: start with FXP)
[AI]
FXP_aifxp4=18154
aiMod4AI2=18038

; This has the details of the Fixedpoint (only FPGA)
[FXP_aifxp4]
Sign=1
Word Length=64
Integer Word Length=0
```

Laboratório Nacional
de Luz Síncrotron

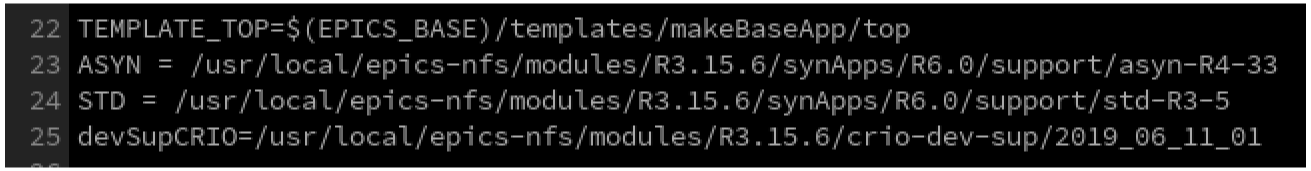
4.2 EPICS IOC – CFG.INI (6/10)



```
[SCALERS]
SCALER_DIGITAL=0
SCALER_ANALOG=1


[SCALER_ANALOG]
Enable=1800E
Gate=18006
OneShot=1800A
Counters=18010
Preset Values=18000
Number of Counters=2
Done=18016

[SCALER_DIGITAL]
Enable=1801E
Gate=18022
OneShot=1801A
Counters=18028
Preset Values=1802C
Number of Counters=2
Done=18026
```

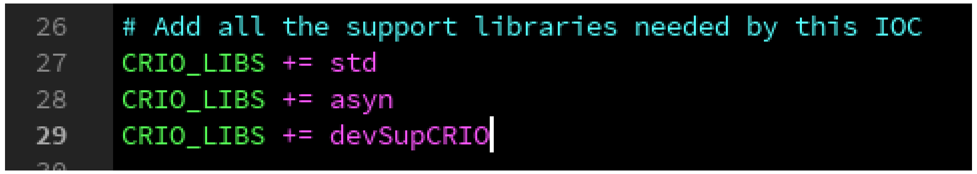
4.2 EPICS IOC – Settings (7/10)

Introduce path of deviceSupportCrio to IOC
$(TOP)/configure/RELEASE



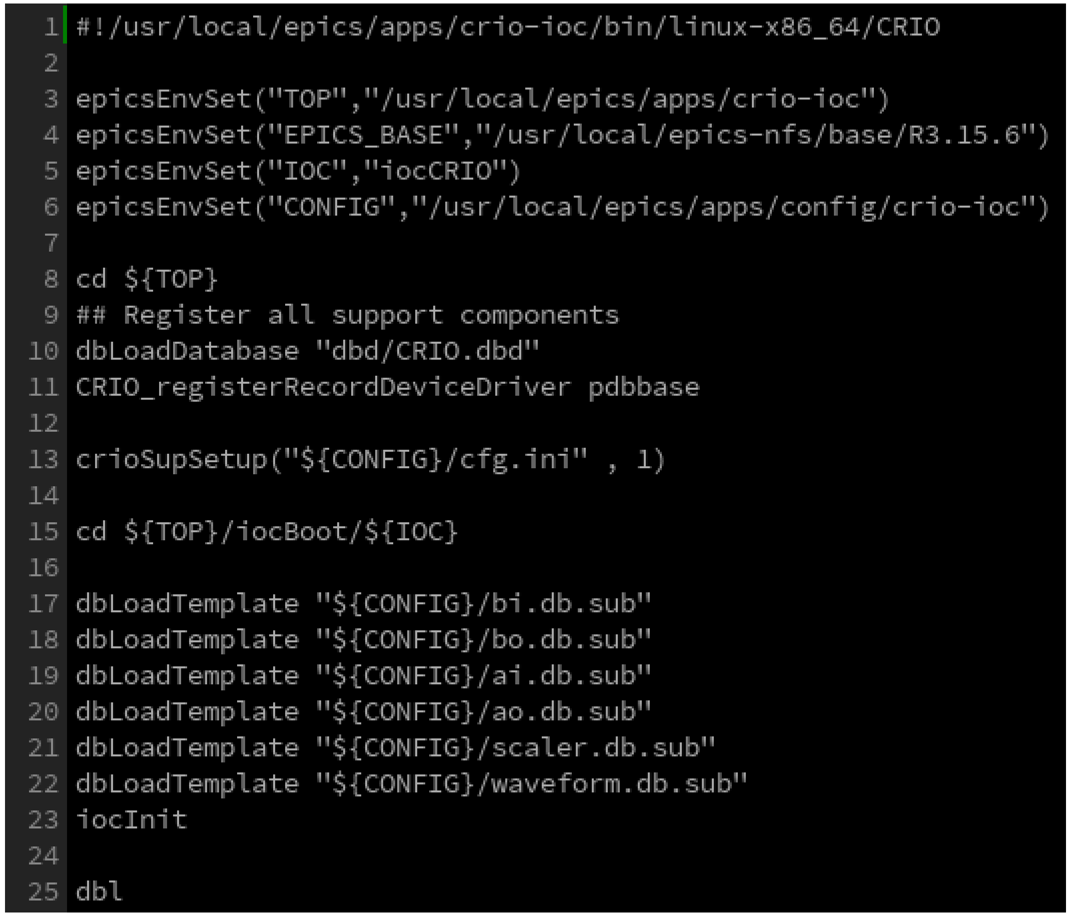
```
TEMPLATE_TOP=$(EPICS_BASE)/templates/makeBaseApp/top
ASYN = /usr/local/epics-nfs/modules/R3.15.6/synApps/R6.0/support/asyn-R4-33
STD = /usr/local/epics-nfs/modules/R3.15.6/synApps/R6.0/support/std-R3-5
devSupCRIO=/usr/local/epics-nfs/modules/R3.15.6/crio-dev-sup/2019_06_11_01
```

Introduce deviceSupportCrio library name to IOC
$(TOP)/CRIOApp/src/Makefile



```
# Add all the support libraries needed by this IOC
CRIO_LIBS += std
CRIO_LIBS += asyn
CRIO_LIBS += devSupCRIO
```

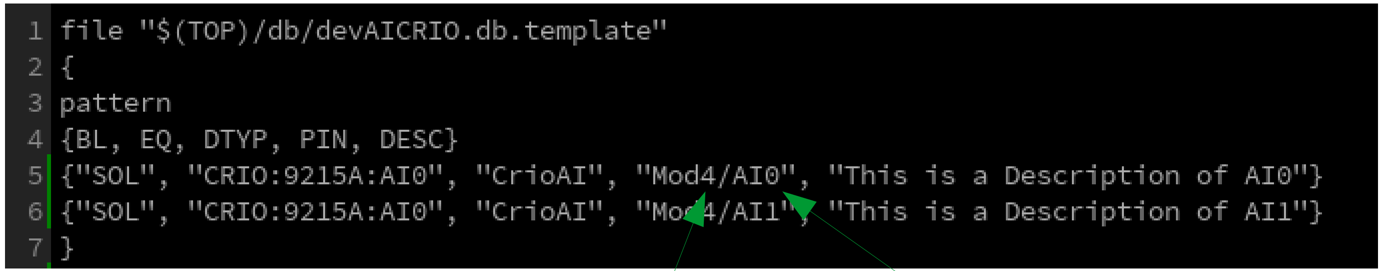
4.2 EPICS IOC – st.cmd (8/10)



```
#!/usr/local/epics/apps/crio-ioc/bin/linux-x86_64/CRIO

epicsEnvSet("TOP","/usr/local/epics/apps/crio-ioc")
epicsEnvSet("EPICS_BASE","/usr/local/epics-nfs/base/R3.15.6")
epicsEnvSet("IOC","iocCRIO")
epicsEnvSet("CONFIG","/usr/local/epics/apps/config/crio-ioc")

cd ${TOP}
## Register all support components
dbLoadDatabase "dbd/CRIO.dbd"
CRIO_registerRecordDeviceDriver pdbbase

crioSupSetup("${CONFIG}/cfg.ini" , 1)

cd ${TOP}/iocBoot/${IOC}

dbLoadTemplate "${CONFIG}/bi.db.sub"
dbLoadTemplate "${CONFIG}/bo.db.sub"
dbLoadTemplate "${CONFIG}/ai.db.sub"
dbLoadTemplate "${CONFIG}/ao.db.sub"
dbLoadTemplate "${CONFIG}/scaler.db.sub"
dbLoadTemplate "${CONFIG}/waveform.db.sub"
iocInit

dbl
```

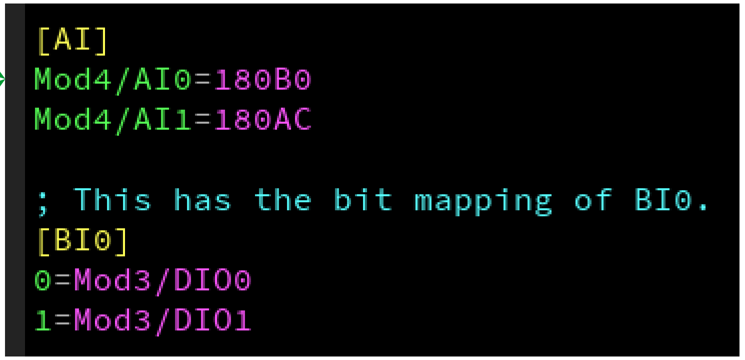
Even the command file does not need any modifications!

4.2 EPICS IOC – templates (9/10)



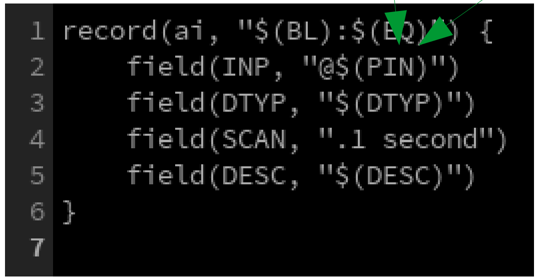
```
file "$(TOP)/db/devAICRIO.db.template"
{
pattern
{BL, EQ, DTYP, PIN, DESC}
{"SOL", "CRIO:9215A:AI0", "CrioAI", "Mod4/AI0", "This is a Description of AI0"}
{"SOL", "CRIO:9215A:AI0", "CrioAI", "Mod4/AI1", "This is a Description of AI1"}
}
```

AI db substitutions



```
[AI]
Mod4/AI0=180B0
Mod4/AI1=180AC

; This has the bit mapping of BI0.
[BI0]
0=Mod3/DIO0
1=Mod3/DIO1
```

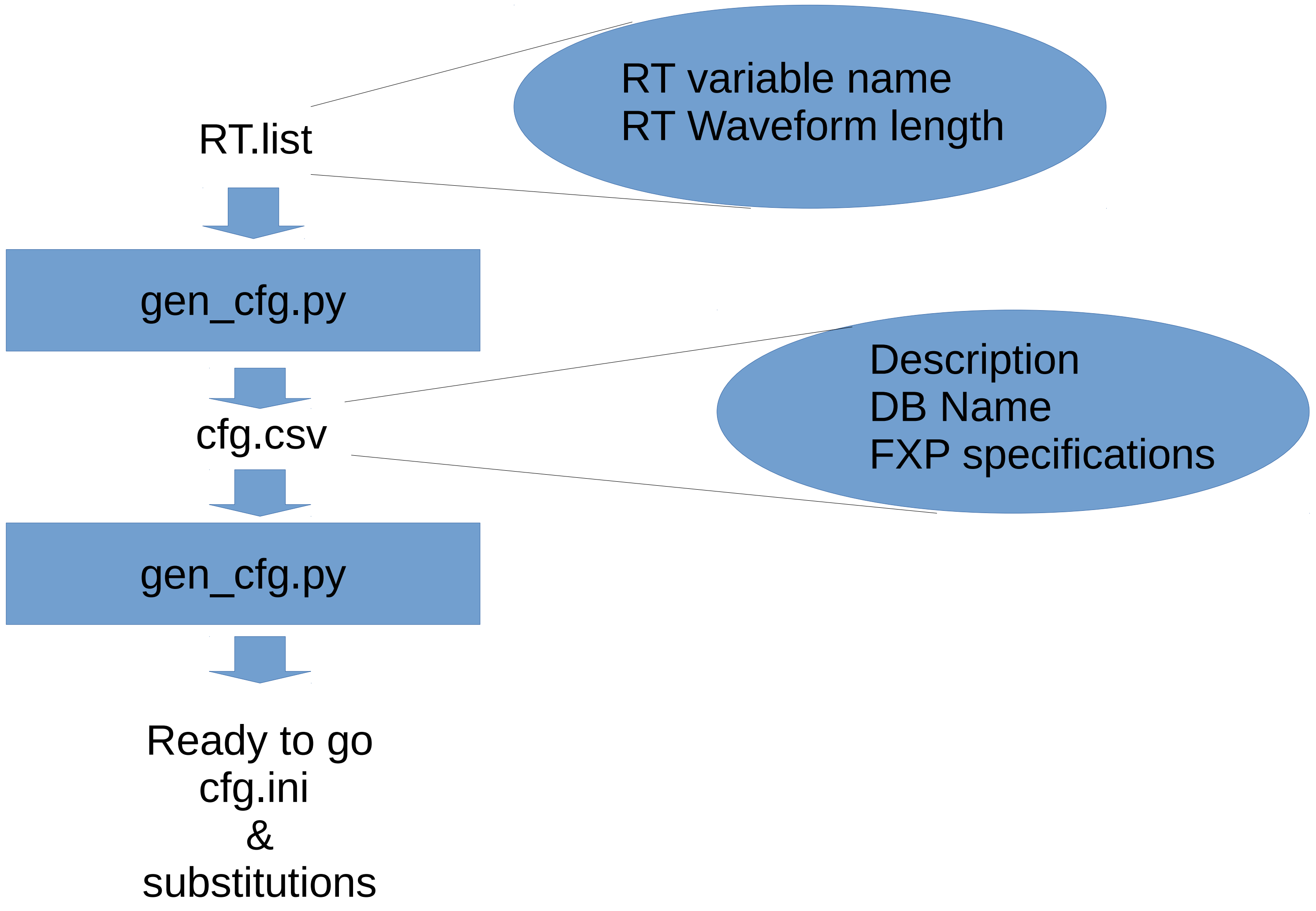
This name must correspond to the name in the cfg.ini file



```
record(ai, "$(BL):$(EQ)") {
    field(INP, "@$(PIN)")
    field(DTYP, "$(DTYP)")
    field(SCAN, ".1 second")
    field(DESC, "$(DESC)")
}
```

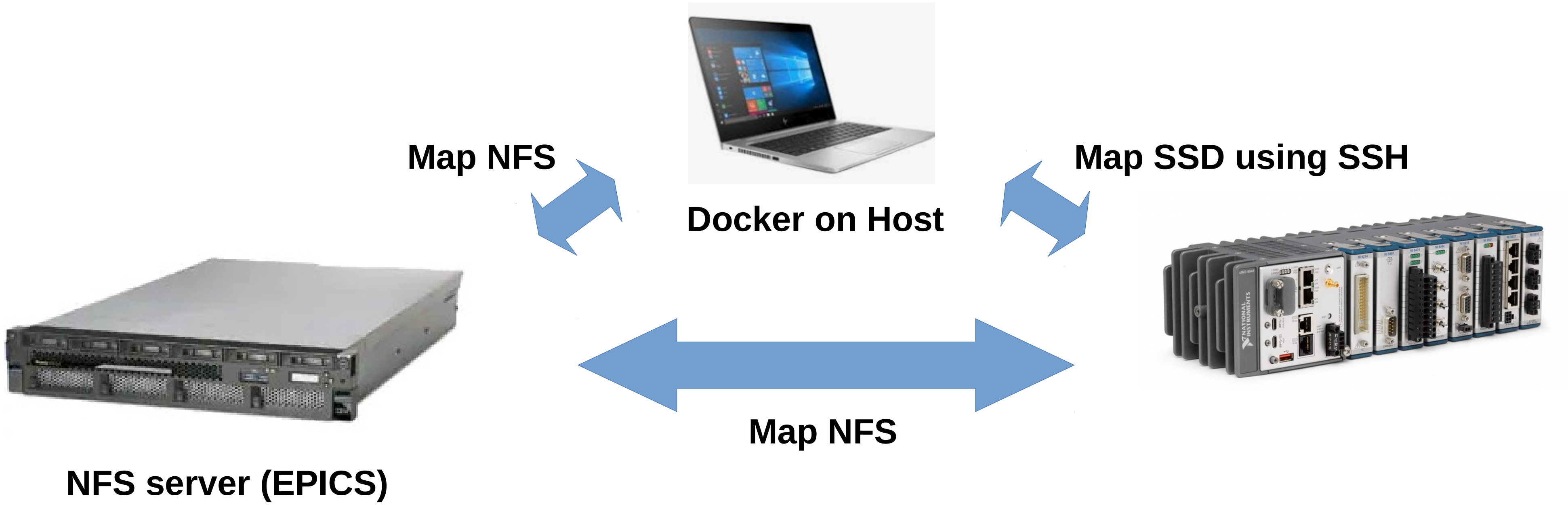
AI db template

4.2 Automatic generation (10/10)



IOC compilation needed?

- We now have a docker compilation environment* that compiles using a processor of another host and using the CRIO SSD



*https://gitlab.cnpem.br/SOL/Docker/dev-crio.git

Softwares used

- EPICS 3.15
- Synapps 6.0 (Scaler)
- Labview 2018
- 2018.5 linuxRT firmware
- Compact RIO 9035/9045

Laboratório Nacional de Luz Síncrotron

- Exception handling
  - Errors in the cfg.ini file, templates, or any inconsistency found appear on the EPICS IOC command prompt
    - e.g. Error on read - [LibCrioLinux] Property [RT_DBL_AI0]: Query returned null.
  - Since exceptions are redirected to epics terminal, the IOC does not stop functioning, so check your messages!

Known limitations

- Binary inputs are limited to 64 bits
- Moving U64, I64 (64-bits) also is lossy since these variables are converted to double in EPICS, and double is 52 bits precision

Nheengatu - Sharepoint

Nheengatu - git repository