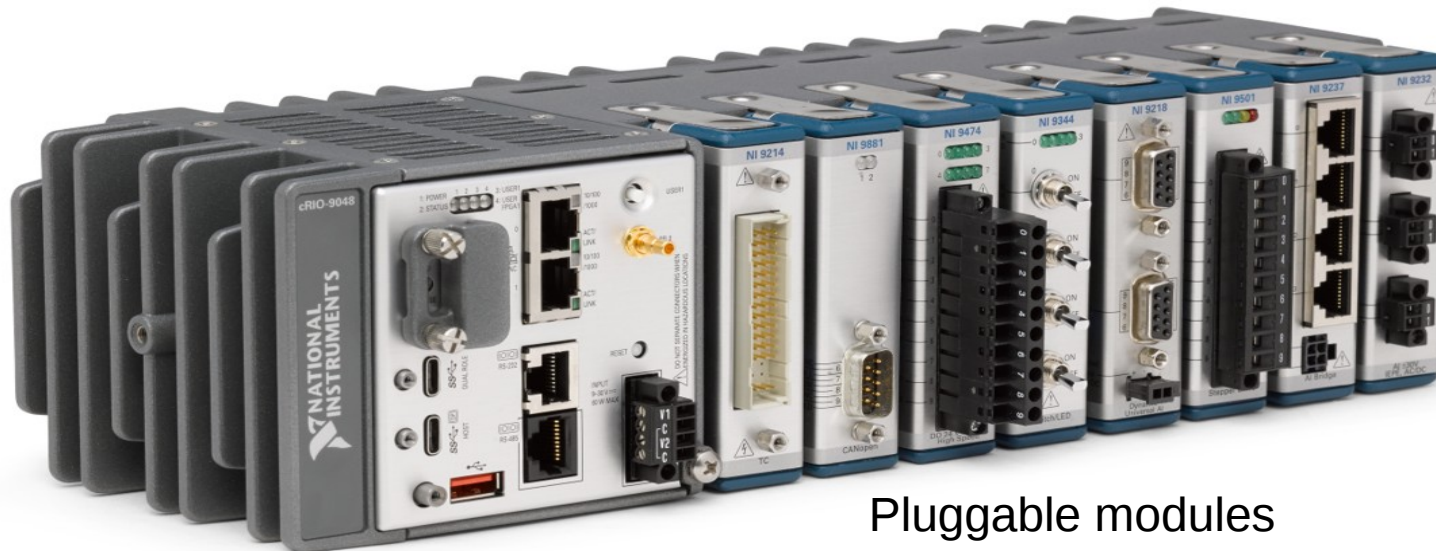# Project Nheengatu: EPICS support for CompactRIO FPGA and LabVIEW-RT
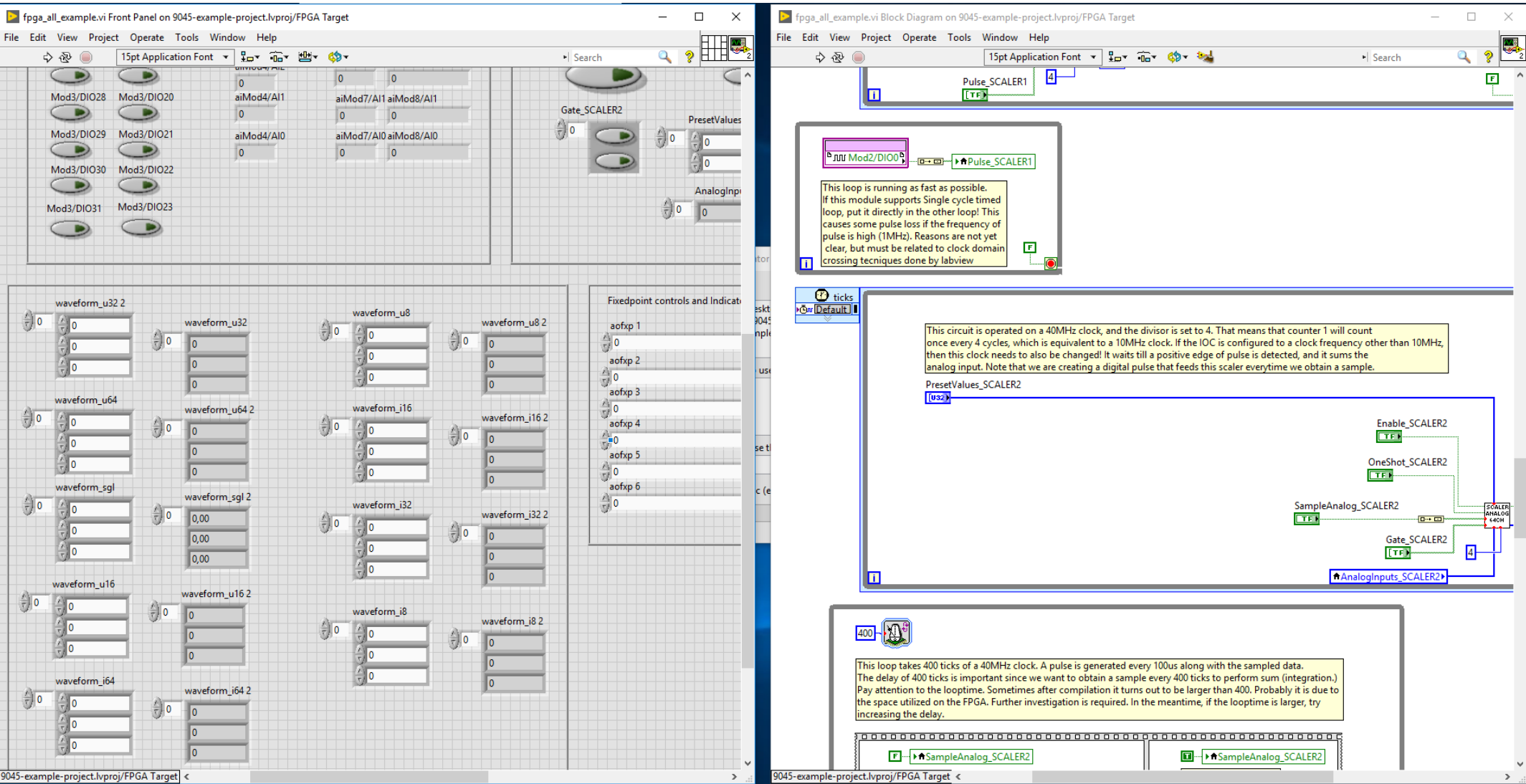
CompactRIO 9045

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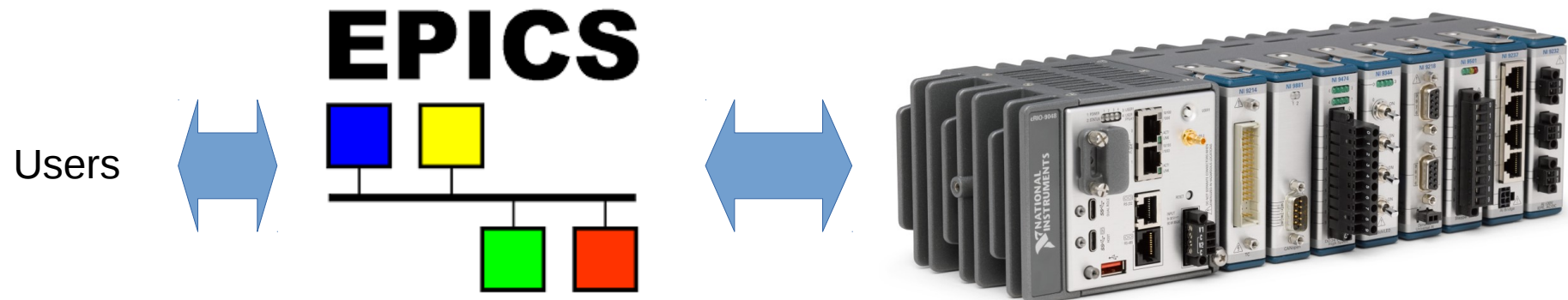
Pluggable modules

LabVIEW

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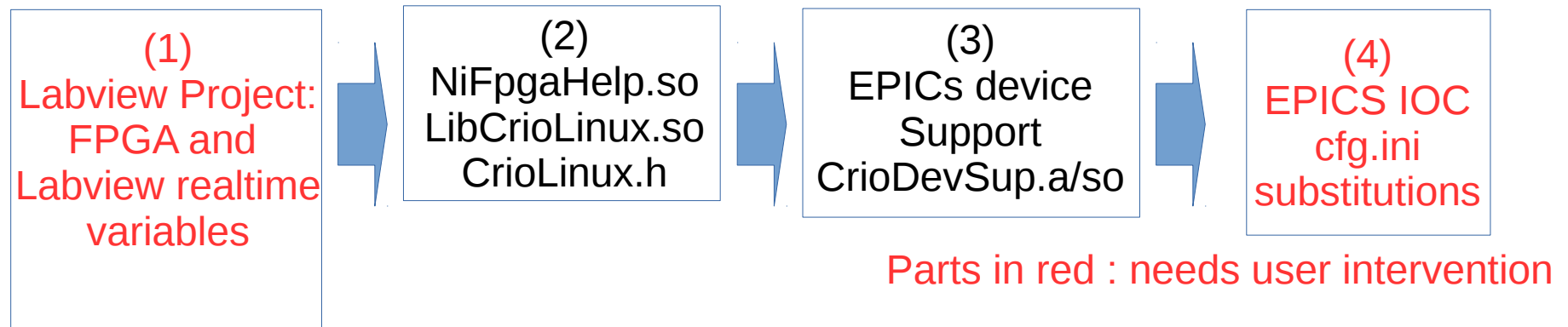
Objective

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

FPGA
LabviewRT (runs in LinuxRT)

Users

Nheengatu architecture

| (1) Labview Project: FPGA and Labview realtime variables	→	(2) NiFpgaHelp.so LibCrioLinux.so CrioLinux.h	→	(3) EPICs device Support CrioDevSup.a/so	→	(4) EPICS IOC cfg.ini substitutions

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
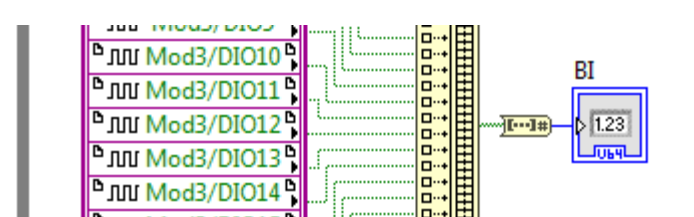   support types

1.1 Handling FPGA variables (1/6)

- **BIs: Concatenate and connect to U64 Indicator**

- **BOs: As is (boolean indicator)**

Binary output : as is
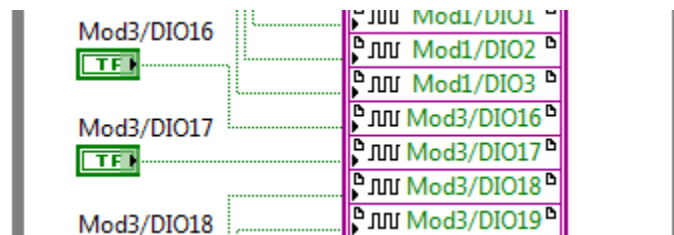
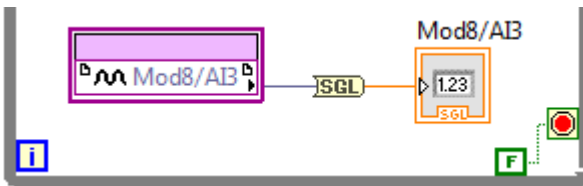**Binary input : Convert to 64-bit U64**
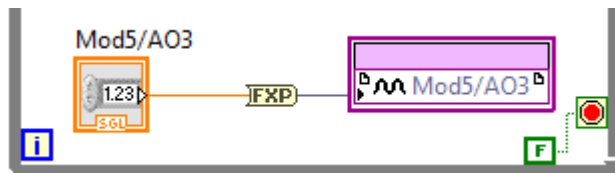
1.1 Handling FPGA variables (2/6)

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
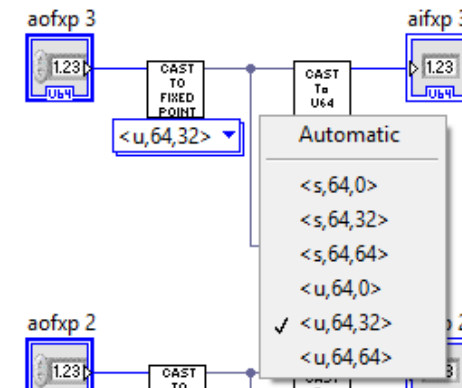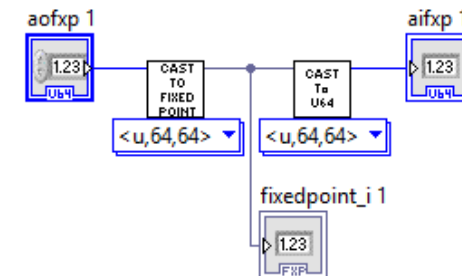
Analog output : use cast to fixed-point polymorphic VI

1.1 FPGA variables (3/6)

Scaler

Scaler digital

Scaler IP in VHDL
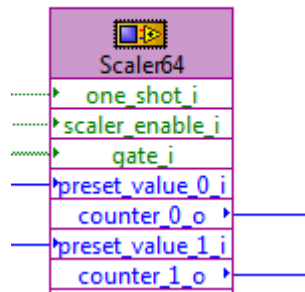
Validated with testbench
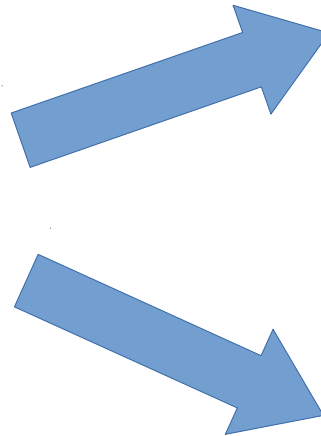
If not all 64 counters are used, the logic can simply be optimized by reducing the size of the fixed input arrays.

FPGA utilization can be reduced by approximately 15% if this optimization is applied
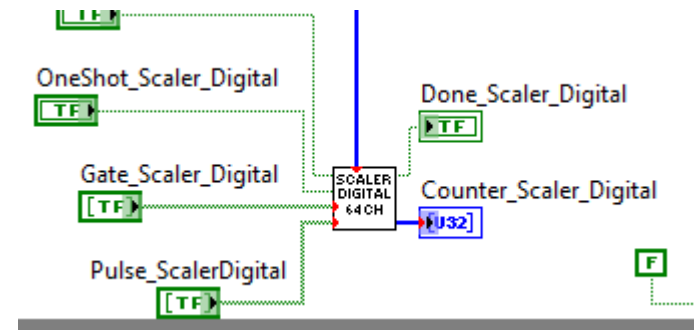
Scaler Analog
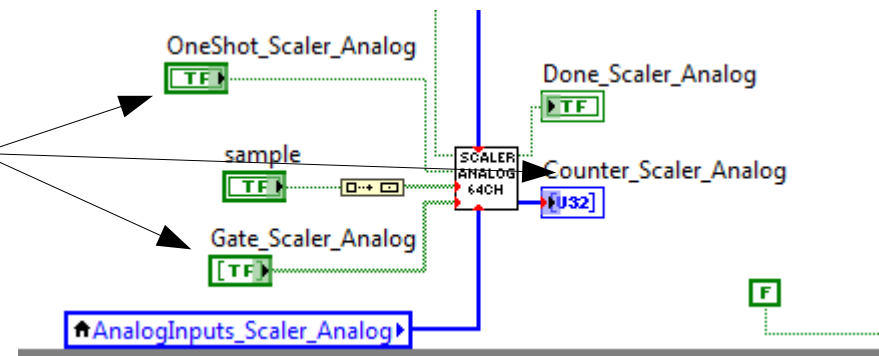
1.1 Handling FPGA variables (4/6)

Reading arrays : instantiate indicator array

- **I08 I16, I32, I64**

- **U08, U16, U32, U64**

- **Single precision floating point**

1.1 FPGA variables (5/6)

Generate bitstream and C API files
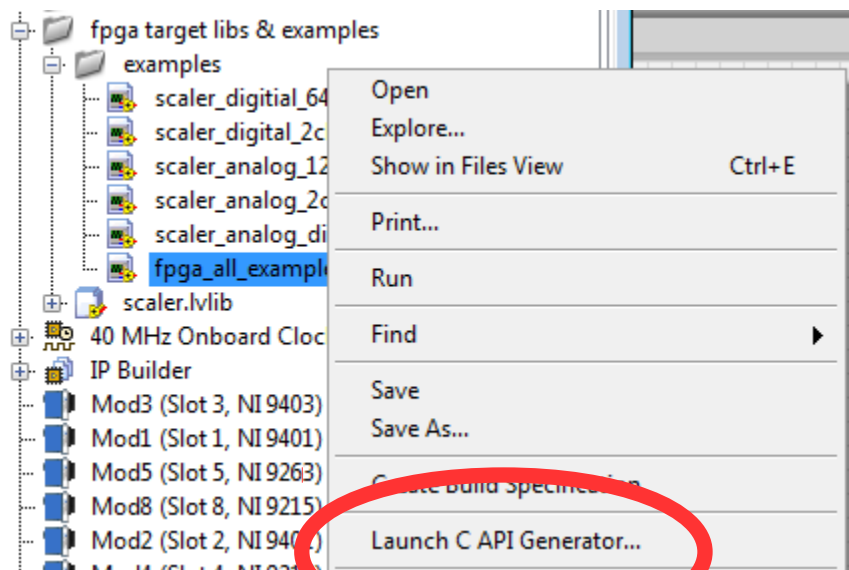
Generates header file with addresses

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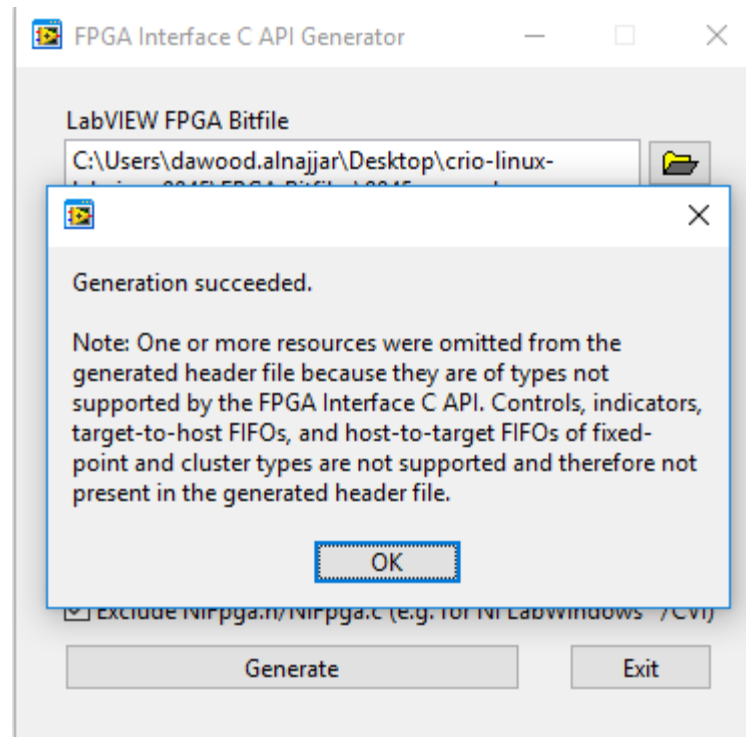
Make sure that non of your indicators and controls are omitted by abiding with the data types defined above!

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.
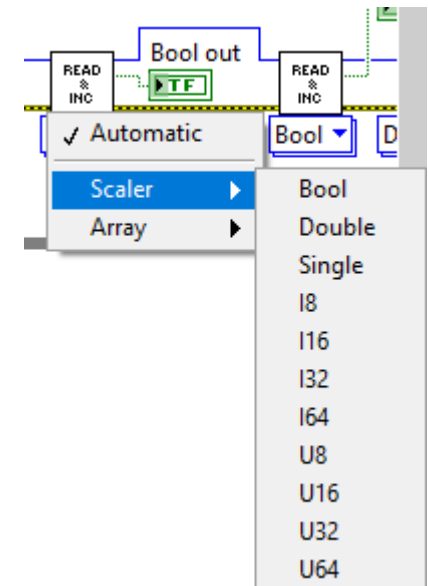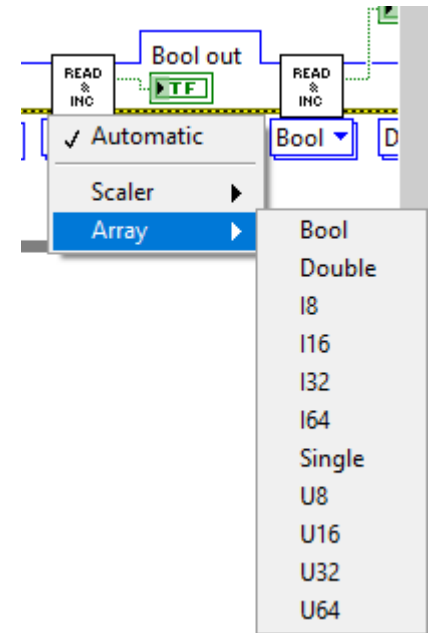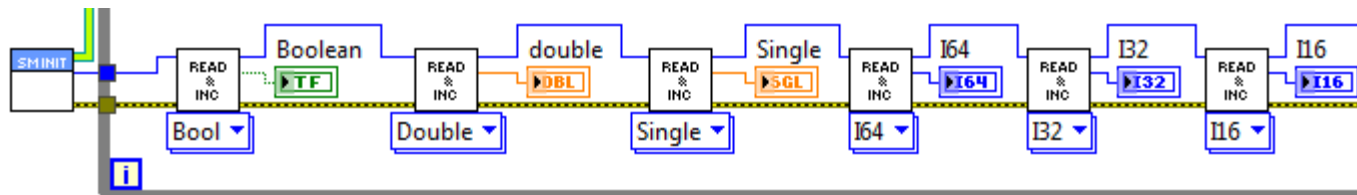
2 & 3 libCrioLinux.so and EPICS device support

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 (1/12)

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

4.2 EPICS IOC – CFG.INI (2/12)


```
; - 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 (3/12)

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

```
56    Mod5/AO0=180B4
57    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 (4/12)

- [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

```
43   [BIAddresses]
44   BI0=180A0
45   RT_BOL_BITest=21
46
47   ; This has the bit mapping of BI0.
48   [BI0]
49   0=Mod3/DIO0
50   1=Mod3/DIO1
51
52
53   ; This has the address of each AO peripheral.
54   ; Category must have name AO.
55   [AO]
56   Mod5/AO0=180B4
57   RT_DBL_AO1=1
58
59   ; This has the address of each AI peripheral. TCs are also
60   ; Considered as AI. Category must have name AI.
61   [AI]
62   Mod4/AI0=180B0
63   RT_DBL_AI0=2
64
65
66   ; This has the address of each BO peripheral.
67   ; Category must have name BO.
68   [BO]
69   Mod1/DIO0=18092
70   RT_BOL_BO0=0
```

4.2 EPICS IOC – CFG.INI (5/12)

- [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
```

4.2 EPICS IOC – CFG.INI (6/12)

```
73    [SCALERS]
74    SCALER_DIGITAL=0
75    SCALER_ANALOG=1
76
77
78    [SCALER_ANALOG]
79    Enable=1800E
80    Gate=18006
81    OneShot=1800A
82    Counters=18010
83    Preset Values=18000
84    Number of Counters=2
85    Done=18016
86
87    [SCALER_DIGITAL]
88    Enable=1801E
89    Gate=18022
90    OneShot=1801A
91    Counters=18028
92    Preset Values=1802C
93    Number of Counters=2
94    Done=18026
95
```

4.2 EPICS IOC – Settings (7/12)

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

```
22  TEMPLATE_TOP=$(EPICS_BASE)/templates/makeBaseApp/top
23  ASYN = /usr/local/epics-nfs/modules/R3.15.6/synApps/R6.0/support/asyn-R4-33
24  STD = /usr/local/epics-nfs/modules/R3.15.6/synApps/R6.0/support/std-R3-5
25  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

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

4.2 EPICS IOC – crio-ioc.cmd (8/12)

```
 1  #!/home/ABTLUS/SOL/crio-ioc/bin/linux-x86_64/CRIO
 2
 3  epicsEnvSet("TOP","/usr/local/epics/apps/crio-ioc")
 4  epicsEnvSet("EPICS_BASE","/usr/local/epics-nfs/base/R3.15.6")
 5  epicsEnvSet("IOC","iocCRIO")
 6  epicsEnvSet("CONFIG","/usr/local/epics/apps/config/crio-ioc")
 7  epicsEnvSet("AUTOSAVE","/opt/autosave")
 8  cd ${TOP}
 9
10  dbLoadDatabase "dbd/CRIO.dbd"
11  CRIO_registerRecordDeviceDriver pdbbase
12  set_requestfile_path($(CONFIG))
13  set_savefile_path($(AUTOSAVE))
14  set_pass1_restoreFile("crioioc.sav", "")
15
16  crioSupSetup("${CONFIG}/cfg.ini" , 1)
17
18  cd ${TOP}/iocBoot/${IOC}
19
20  dbLoadTemplate "${CONFIG}/bi.db.sub"
21  dbLoadTemplate "${CONFIG}/bo.db.sub"
22  dbLoadTemplate "${CONFIG}/ai.db.sub"
23  dbLoadTemplate "${CONFIG}/ao.db.sub"
24  dbLoadTemplate "${CONFIG}/scaler.db.sub"
25  dbLoadTemplate "${CONFIG}/waveform.db.sub"
26  iocInit
27
28  < "$(CONFIG)/init-pv.cmd"
29
30  create_monitor_set("crioioc.req", 1, "")
31
32  dbl
```

Even the command file does not need any modifications!

4.2 EPICS IOC – templates (9/12)

```
1 file "$(TOP)/db/devAICRIO.db.template"
2 {
3 pattern
4 {BL, EQ, DTYP, PIN, DESC}
5 {"SOL", "CRIO:9215A:AI0", "CrioAI", "Mod4/AI0", "This is a Description of AI0"}
6 {"SOL", "CRIO:9215A:AI0", "CrioAI", "Mod4/AI1", "This is a Description of AI1"}
7 }
```

AI db substitutions

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

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

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

```
1 record(ai, "$(BL):$(EQ)") {
2     field(INP, "@$(PIN)")
3     field(DTYP, "$(DTYP)")
4     field(SCAN, ".1 second")
5     field(DESC, "$(DESC)")
6 }
7
```
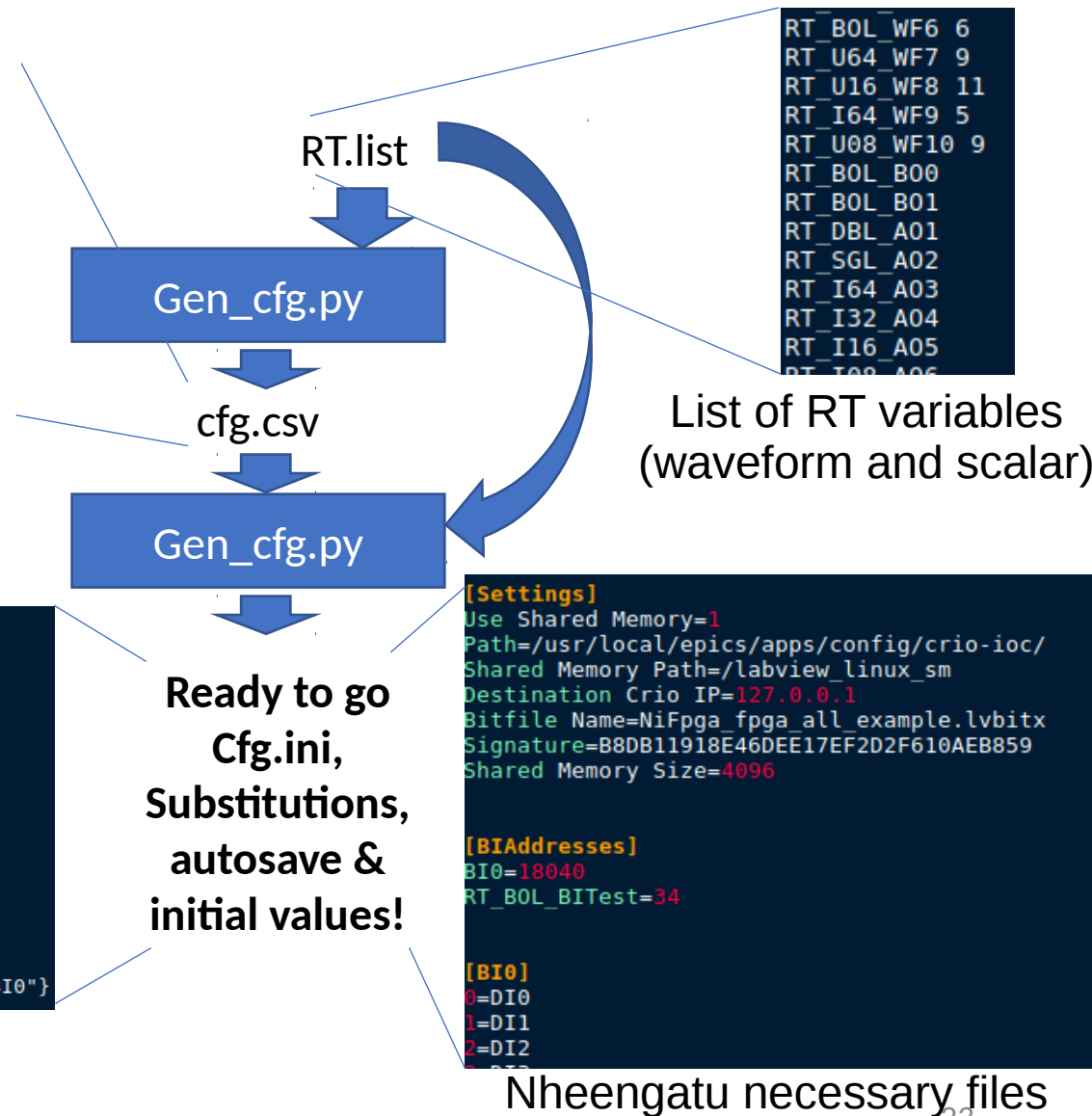
AI db template

4.2 Automatic generation (10/12)

	BI INI NAME	BI DB NAME	BI DESCRIPTION
	RT_BOL_BITest	RT:BI0	This is a Description of BI0
	DI0	9403A:BI0	This is a Description of BI1
	DI1	9403A:BI1	This is a Description of BI2
	DI2	9403A:BI2	This is a Description of BI3
	DI3	9403A:BI3	This is a Description of BI4
	DI4	9403A:BI4	This is a Description of BI5
	DI5	9403A:BI5	This is a Description of BI6
	DI6	9403A:BI6	This is a Description of BI7
	DI7	9403A:BI7	This is a Description of BI8

List of all variables
(Needs to be filled by user)

RT.list

Gen_cfg.py

cfg.csv

Gen_cfg.py

```
RT_BOL_WF6 6
RT_U64_WF7 9
RT_U16_WF8 11
RT_I64_WF9 5
RT_U08_WF10 9
RT_BOL_B00
RT_BOL_B01
RT_DBL_A01
RT_SGL_A02
RT_I64_A03
RT_I32_A04
RT_I16_A05
```

List of RT variables
(waveform and scalar)

**Ready to go
Cfg.ini,
Substitutions,
autosave &
initial values!**

```
file "$(TOP)/db/devBICRIO.db.template"
{
pattern
{BL, EQ, DTYP, PIN, DESC}
{"SOL", "CRIO2:9403A:BI0", "CrioBI", "DI0", "This is a Description of BI1"}
{"SOL", "CRIO2:9403A:BI2", "CrioBI", "DI2", "This is a Description of BI3"}
{"SOL", "CRIO2:9403A:BI19", "CrioBI", "DI19", "This is a Description of BI20"}
{"SOL", "CRIO2:9403A:BI22", "CrioBI", "DI22", "This is a Description of BI23"}
{"SOL", "CRIO2:9403A:BI10", "CrioBI", "DI10", "This is a Description of BI11"}
{"SOL", "CRIO2:9403A:BI18", "CrioBI", "DI18", "This is a Description of BI19"}
{"SOL", "CRIO2:9403A:BI13", "CrioBI", "DI13", "This is a Description of BI14"}
{"SOL", "CRIO2:9403A:BI16", "CrioBI", "DI16", "This is a Description of BI17"}
{"SOL", "CRIO2:9403A:BI8", "CrioBI", "DI8", "This is a Description of BI9"}
{"SOL", "CRIO2:9403A:BI14", "CrioBI", "DI14", "This is a Description of BI15"}
{"SOL", "CRIO2:9403A:BI11", "CrioBI", "DI11", "This is a Description of BI12"}
{"SOL", "CRIO2:9403A:BI4", "CrioBI", "DI4", "This is a Description of BI5"}
{"SOL", "CRIO2:RT:BI0", "CrioBI", "RT_BOL_BITest", "This is a Description of BI0"}
{"SOL", "CRIO2:9403A:BI12", "CrioBI", "DI12", "This is a Description of BI13"}
```

IOC necessary files

```
[Settings]
Use Shared Memory=1
Path=/usr/local/epics/apps/config/crio-ioc/
Shared Memory Path=/labview_linux_sm
Destination Crio IP=127.0.0.1
Bitfile Name=NiFpga_fpga_all_example.lvbitx
Signature=B8DB11918E46DEE17EF2D2F610AEB859
Shared Memory Size=4096


[BIAddresses]
BI0=18040
RT_BOL_BITest=34


[BI0]
0=DI0
1=DI1
2=DI2
```

Nheengatu necessary files

4.2 EPICS IOC – templates (11/12)

usage: usage: gen_cfg.py [-h] [-u] [-d DST] [-p PATH] [-s SRC] [--binum BINUM]
*   [--extract] [--ip IP] [--smfname SMFNAME] [--smsize SMSIZE]*
*   [--aikey AIKEY] [--aokey AOKEY] [--bokey BOKEY]*
*   [--bikey BIKEY] [--fxpkey FXPKEY] [--scalerkey SCALERKEY]*
*   [--waveformkey WAVEFORMKEY] [--beamline BEAMLINE]*
*   [--bidtyp BIDTYP] [--aidtyp AIDTYP] [--bodtyp BODTYP]*
*   [--aodtyp AODTYP] [--wfdtyp WFDTYP] [--crio CRIO]*
*   [--scalerdtyp SCALERDTYP] [--cfgcsv CFGCSV]*
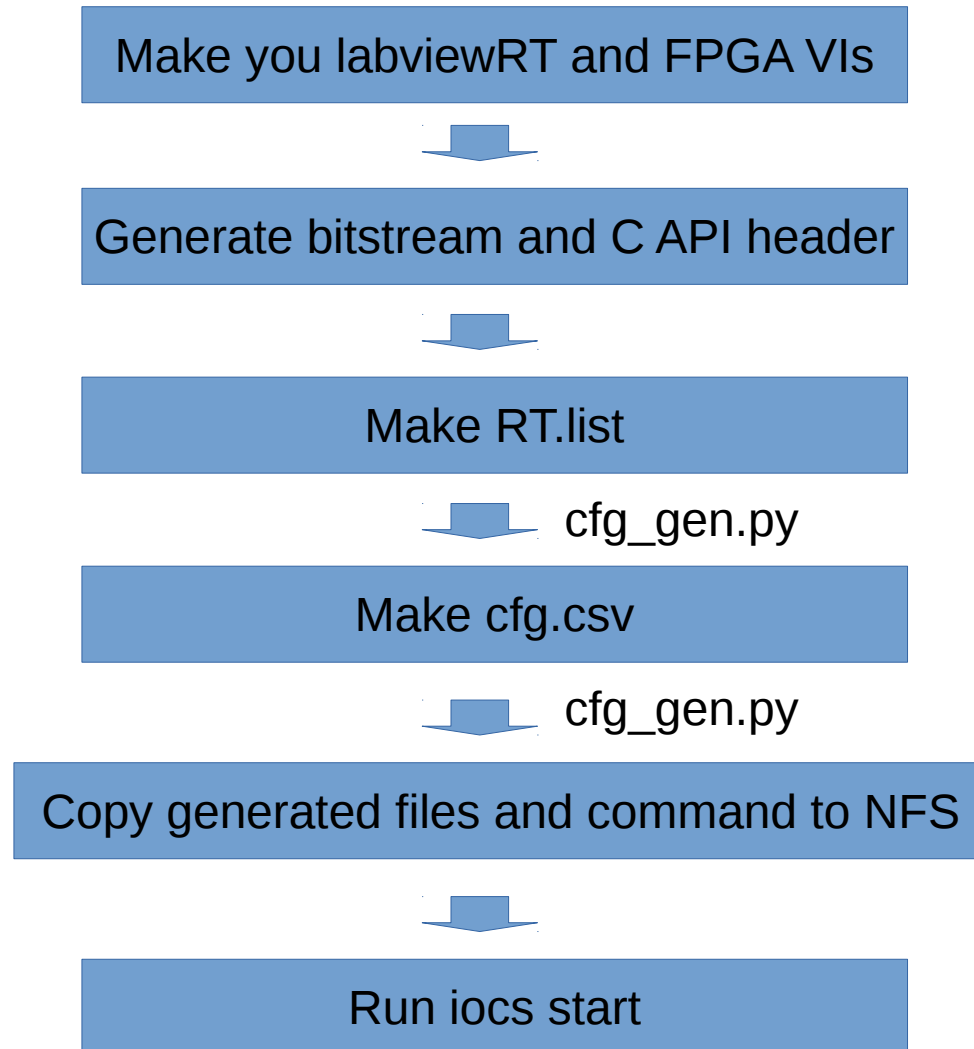
For help : ./gen_cfg.py -h

4.2 Generated folder (12/12)

Generate **reference** folder that has all files necessary to reproduce that specific setup

```
[dawood.alnajjar@nfs-epics config]$ tree crio-ioc
crio-ioc
├── ai.db.sub
├── ao.db.sub
├── bi.db.sub
├── bo.db.sub
├── cfg.ini
├── crioioc.req
├── init-pv.cmd
├── NiFpga_fpga_all_example.lvbitx
├── reference
│   ├── cfg.csv
│   ├── command.sh
│   ├── reference.h
│   └── RT.list
├── scaler.db.sub
└── waveform.db.sub

1 directory, 14 files
```
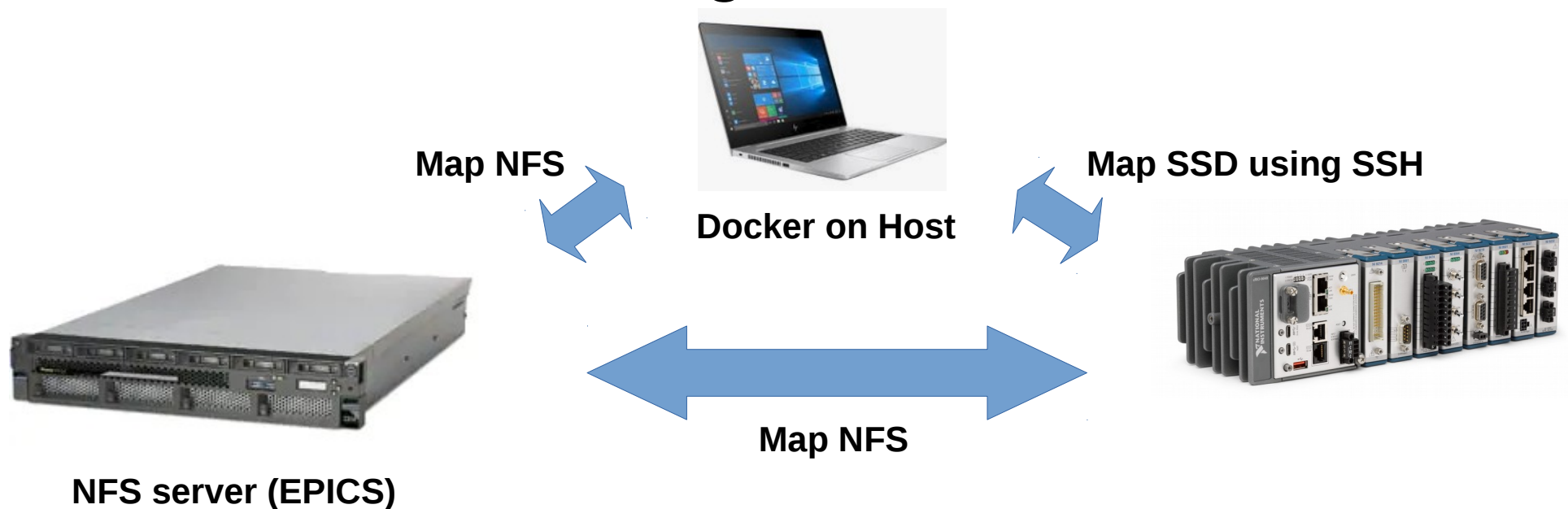
Big picture

Make you labviewRT and FPGA VIs

Generate bitstream and C API header

Make RT.list

cfg_gen.py

Make cfg.csv

cfg_gen.py

Copy generated files and command to NFS

Run iocs start

NFS setup-bl folder

```
[dawood.alnajjar@nfs-epics SOL]$ tree CRIO4/
CRIO4/
└── epics
    ├── apps
    │   ├── config
    │   │   ├── crio-ioc
    │   │   │   ├── ai.db.sub
    │   │   │   ├── ao.db.sub
    │   │   │   ├── bi.db.sub
    │   │   │   ├── bo.db.sub
    │   │   │   ├── cfg.ini
    │   │   │   ├── crioioc.req
    │   │   │   ├── init-pv.cmd
    │   │   │   ├── NiFpga_fpga_all_example.lvbitx
    │   │   │   ├── reference
    │   │   │   │   ├── cfg.csv
    │   │   │   │   ├── command.sh
    │   │   │   │   ├── reference.h
    │   │   │   │   └── RT.list
    │   │   │   ├── scaler.db.sub
    │   │   │   └── waveform.db.sub
    │   │   └── crio-ioc.cmd
    │   └── crio-ioc -> /usr/local/epics-nfs/apps/crio-ioc/2019_06_25_01/
    └── base -> /usr/local/epics-nfs/base/R3.15.6/
```

IOC compilation needed?

- Docker compilation environment* that compiles using a processor of another host and using the CRIO SSD for testing on the CRIO

Map NFS

Docker on Host

Map SSD using SSH

Map NFS

NFS server (EPICS)

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

Softwares used

- EPICS 3.15

- Synapps 6.0 (Scaler, Autosave)

- Labview 2018

- 2018.5 linuxRT firmware

- Compact RIO 9035/9045

- 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.
  - Non-critical exceptions are redirected to epics terminal, and 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

- Number of indicators and controls on FPGA is limited due to available resources

Nheengatu - Sharepoint
Nheengatu - git repository