ConfigDB from YAML

Conversation with Matt/Ric/Riccardo/cpo on April 4, 2024. For complex area detectors.

WE CAN REQUEST CHANGES FROM TID

You could consider running Ric's scripts and create a dummy epixM in
configdb (don't touch the official one).

In epixuhr_config_store can we create a Root object and iterate over
it to create the json? Worth exploring. Might require the hardware
to exist? Or we could use "simulation" somehow?

****

- look in yaml files
- translate those yaml files into epixuhr_config_store.py
  (do with python?)
  o another possibility: rogue command to dump out the registers?
    pick out the read-write ones. might also include the type info
- all these registers should go into the "expert" section
  o there is also a "user" section with a simplified user variables
    which get "translated" by code into the official "expert" structure.
    Ric says there are "helper" variables in the user section,
    e.g. charge-injection patterns, that aren't necessarily modified
    by the user. The charge-injection script modifies those variables.
    Ric says perhaps this should be in a third section? (not user/expert?)
- need to find out the types by looking at devGui or firmware/*.py files
  that define devGui
- set most values to 0 then tweak critical ones to non-zero
- a command for the next step:
    (first command creates confidb schema)
  o python epixm320_config_store.py --prod --user tstopr --inst tst --alias BEAM --name epixm --segm 0 --id epixm320_serial1234 --dir /cds/home/c/cpo/git/epix-hr-m-320k/
    (second command fills in values)
  o python epixhr_config_from_yaml.py --prod --user tstopr --inst tst --alias BEAM --name epixm --segm 0 --id epixm320_serial1234 --yaml Root:/cds/home/c/cpo/git/epix-hr-m-320k/software/config/ePixHRM320k_75000018efb4ab01_ASIC_u1.yml
  "--dir" is used by the *_config_store.py to locate any files that may be needed. For the ePixM, these are csv files in the config/ area for setting up the PLL. The UHR doesn't seem to have .csv files, but it may be useful for picking up the .txt files in config/pll.

    top = epixm320_cdict(args.dir+'/software/config')
    top.setInfo('epixm320hw', args.name, args.segm, args.id, 'No comment')

these are the csv file-loading lines:

    top.set(base+'_250_MHz', np.loadtxt(prjCfg+'/EPixHRM320KPllConfig250Mhz.csv', dtype='uint16', delimiter=',', skiprows=1, converters=conv))
    top.set(base+'_125_MHz', np.loadtxt(prjCfg+'/EPixHRM320KPllConfig125Mhz.csv', dtype='uint16', delimiter=',', skiprows=1, converters=conv))

csv are passed to the root.fnInitAsic(dev, cmd, (csv_file_index_into_list_of_csv_files,)). For the UHR, this function is in the root.App block.

run epixhr_config_from_yaml.py with different arguments to update the values in the schema. epixm has 5 or 6 yaml files. These are the lines and files for the UHR (see _App.py: fnInitAsic()):

```
            self.filenameRegisterControl   = "config/UhrWaveformControlAxi_registers.yml"
            self.filenameTriggerReg        = "config/UhrTrigControlAxi_registers.yml"
            self.filenameSACIReg           = "config/ePixUHR_SACI_Registers.yml"
            self.filenameFramerReg         = "config/UhrFramerAxi_registers.yml"
            self.filenameGeneral           = "config/ePixUHR_camera_general_settings.yml"
```

new yaml files (with a new "schema") can be provided by the TID group (and the detector group?) as time goes on. Ric has a mechanism to update the schema but preserve the values (!) (see https://confluence.slac.stanford.edu/display/PSDMInternal/Debugging+DAQ#DebuggingDAQ-MakingSchemaUpdatesinconfigdb).
Can add/drop/modify fields as necessary with Ric's sheme.

in epixm_config.py reads the configdb and re-creates the .yaml files in /tmp (so that changes made using the configDb editor are also picked up). Also creates Root.filename<yamlname> variables. yamlname is the third argument here (PowerSupply):

    toYaml('App',['PowerControl'],'PowerSupply')

then this InitASIC gets the /tmp/yamlname inserting a filename into a variable of this Root class (editing the class): (filename is not passed in; see `dictToYaml()` in `configdb/det_config.py`: This line does the trick: `setattr(dev,'filename'+name,fn)`)
(arg has the index that points to an entry in a list that has clock frequencies, .csv files, etc.)

    cbase.fnInitAsicScript(None,None,arg)

goal might be to get the above script to work. fnInitAsic() itself isn't called by *_config.py because it hardcodes the filename and path.

in future perhaps consider adding arguments instead of modifying class?

****

copy lines from devGui, e.g.

```
cbase = ePixM.Root(top_level      = '/tmp',
          dev                = '/dev/datadev_0',
          pollEn              = False,
          initRead            = False,
          justCtrl            = True,
          fullRateDataReceiverEn = False)
cbase.saveAddressMap("filename")
```

parse with python to generate a schema?

Root python class knows the schema, .yaml has the values

Matt's epixhr_config_from_yaml_set.py iterates over multiple yaml files

****

Watch out for:
- enum keywords have [] () that are invalid python/XTC2 names. can request
  to have them changed, or need to "translate"
- previously debugging those errors in JsonToXtc was difficult, but
  this may be improved with recent changes from Gabriel. Ric thinks
  this may not cover much of the phase-space for errors, but we can
  consider improving it by talking with Gabriel. e.g. a new
  epixm release removed a variable from the yaml, but was in the
  schema and python class. Got a default value of zero and JsonToXtc was unhappy because it expects one of the enum keywords.