Instructions for CORDFER

CORDFER (**C**RIB-**O**rganized **RDF** **E**ncoder to **R**oot), is the code which can be used to convert rdf files to root. This instruction will show how to use and modify the CORDFER according to your own experimental setup.

1 HOW TO INSTALL AND USE CORDFER

Now, CORDFER is only supported on the LINUX system. To install CORDFER, the required softwares are only **root and yaml**. Root provides the necessary libs, and yaml must be present for the recipe file (see the next Sec.2 for detail). Please see the website https://github.com/jbeder/yaml-cpp for yaml installation.

With the head files (@"./include/"), .cpp files (@"./src"), as well as makefile copied from the J1-PC, CORDFER can be easily compiled by typing "make".

Some other directories and files are also required in addition to **"./include/"** , **"./src"** and **makefile** (all of them should be in the same directory level):

"recipe.yaml": see Sec.2;

"./cordfer_map/": contain all the map files required by CORDFER (see Sec.3);

"./rdf/": the raw rdf files should be linked to this directory;

"./rootfile/": contain the root files to be produced.

To execute CORDFER, just type "./cordfer run_number (the run you want to convert)". You can write a batch script, or add a run number loop into the "./src/main.cpp", to make a batch conversion. Please keep in mind that you need to re-make if you modify the files in "./include/" and "./src/".

2 RECIPE

"recipe.yaml" is the most important file that you need to modify according to your own experiment. And the CORDFER will create the root branches and leaves according to the recipe file.

The recipe file is written in YAML format, which is based on the yaml-cpp 0.5.3 to run including a header file "yaml.h".

A sample of a regular recipe.yaml file is present as below.

==========**Sample of recipe file**==========

```
tree name: "tree"
  setup:
  - segment no.: 1 #"RF&F1"
  dtype: Raw
  mapfile: No
  branches:
  - name: [fRf,1,2,S,0]
  - name: [fF1ppac,1,5,S,3]
```

- segment no.: 2 #"PPACs"
dtype: Raw
mapfile: No
branches:
- name: [fPpac,3,5,S,0]
- segment no.: 4 #"CoinReg"
dtype: Bit
mapfile: No
branches:
- name: [fCr,1,3,S,0]
- segment no.: 6 #"CAMAC ADC"
dtype: LeCroy
mapfile: psd.map
branches:
- name: [fPsdE,2,16,S,1]
- segment no.: 8 #"VME ADC"
dtype: CAEN
mapfile: psd_vme_adc.map
branches:
- name: [fF2SsdE,1,1,S,22]
- name: [fSsdE,1,6,S,23]
- name: [fHpsdE,6,5,S,101]
- segment no.: 9 #"VME TDC"
dtype: CAEN
mapfile: psd_vme_tdc.map
branches:
- name: [fSsdT,1,6,S,23]
- name: [fPsdT,2,16,S,31]
- name: [fHpsdT1,1,1,S,105]
- name: [fHpsdT2,1,1,S,110]
- name: [fHpsdT3,1,1,S,115]
- name: [fHpsdT4,1,1,S,120]
- name: [fHpsdT5,1,1,S,125]
- name: [fHpsdT6,1,1,S,130]
==========**End of recipe file**==========

Description:

1. **tree name**: Define the tree name of the root file to be produced. In this sample, the tree name is set as "tree".

2. **segment no**: segment ID. The ID for each segment has been fixed in the DAQ programme. If you want to add a new segment, please check the DAQ code for the segment ID.

3. **dtype**: decoder type for this segment. There are four types: Raw, Bit, LeCroy and CAEN. Please select the decoder type according to the DAQ module used for this segment to decode the raw data.

4. **mapfile**: map file name. Only the CAMAC ADC and VME A/TDC require the map files. Please see the detailed information for the mapping in the next section.

5. **name**: define all the branch information to be created in the root file. There are five parameters for each branch: **1.** branch name; **2.** dimension of the array corresponding to this branch; **3.** channel numbers of each dimension; **4.** data type of this branch, here, "S" indicates short; **5.** ID of the **FIRST** channel according to the map file.

Notes:

1. Basically, you do not need to modify the **Segments 1, 2 and 4**, which are based on the setup of CRIB. Please just update the information of **Segments 6, 8 and 9** according to your experimental setup.

2. If you want to use an array to contain all the leaves for a branch, you must make sure that **IDs of these leaves are continuous** in the map file. Take the HPSD timing signal (segment ID 9) in the sample as an example. There are 6 HPSD timing channels, and according to the definition of the TDC map file, the channel IDs of these signals are 105, 110, 115, 120, 125 and 130, respectively, which are not continuous. Thus you have to define them as branches separately.

3. All the data types defined in CORDFER are short. So please keep the "**S**" as the data type in the recipe.

4. The maximum number of branch for each segment is set to be 9. And the maximum numbers of array dimension, as well as channel for each dimension are set to be 9 and 96, respectively. You can modify these default values in the head file of "./include/FillBranch.h".

5. CORDFER reads the rdf files from the directory of "./rdf/". So please link your rdf files to this directory. The default output path is "./rootfile/". These pathes can be modified in the subroutine of "ReadRecipe" in the head file of "./include/FillBranch.h".

3 MAPPING

The psd.map (which is used for the CAMAC ADC) can be used directly by CORDFER. Please copy it to the directory of "./cordfer_map/".

The psd_vme.map, however, is needed to convert to an appropriate format, which then can be accepted by CORDFER. You can use the code **MapConvert.cpp** to fulfill this conversion. With this small code, the psd_vme.map (the path to this map should be specified in the code) will be separated into psd_vme_adc.map and psd_vme_tdc.map, and stored in the directory of "./cordfer_map/" by default. Then CORDFER will read in these map files (psd_vme_adc.map, psd_vme_tdc.map, as well as psd.map) from "./cordfer_map/".

So please make sure that all these maps are stored in this directory. Of course you can modify the path as you want, but then please remember to update the path information of the maps in CORDFER (the subroutine of "ReadRecipe" in the head file of "./include/FillBranch.h").

4 SHELL SCRIPTS

You can also use shell scripts to set the links. The path to "./bin" directory is needed to be specified by adding the following sentence into your shell profile, i.e., bashrc.

export PATH=$HOME/[path to cordfer directory]/bin:$PATH

Then the following commands can be used.

♣ **setmap**

setmap can make map files from anapaw map files. For instance, if you have map files in the "home/exp/crib/be7_thm/src/", use setmap as

setmap home/exp/crib/be7_thm/src/

and it automatically creates "cordfer_map" with a symbolic link, "@map", by using a perl script.

♣ **setrdf**

setrdf simply creates a symbolic link, "@rdf", where rdf files are. Just use it as setmap.

♣ **cordfer**

It is almost same as "./cordfer" but gives you some information on its usage and version.

5 CALIBRATION

The root file produced by CORDFER only contains the raw data, without calibration. A sample of calibration code is provided in the directory of "./cali_convert/". In this directory, please type "./caliconvert", and then input the minimum and maximum run numbers you want to calibrate.

With this code, the CRIB beam line detectors, i.e., F1-, F2- and F3-PPACs, as well as the RF signals, can be calibrated (the original **ppac.prm** file is required). And you can also add the calibration parameters of your own detectors to this code. The default output path of the calibrated root file is "./cali_rootfile/".