Transmit Multiplexer

The project contains the replacement for the original dechannelizer

FOR UBUNTU DOCKER BUILDS

NOTE: This only works for C builds for docker base

Locally:

- clone this repo
- run the ubuntu build script
- you will need to pass in a tag argument

Pipelines:

- you will need a working pipeline
- you have to change the .gitlab-ci.yml to use the ubuntu docker build script

Build Status

Master

Build Status: pipeline status

Develop

Build Status: pipeline status

Branching Strategy

There are two protected branches:

- master: only people with master level permissions and above can push to here
- develop: only people with master level permissions and above can push to here

The General strategy is as follows:

- Create your feature branch from `develop`
- Name your feature branch with the following syntax `feature/<name_of_feature_being_worked>`
- Make your changes to the code related to dechannelizer and push to your feature branch
- When your feature is finished make a merge request for your feature branch to be merged into develop, assign the POC associated with that feature to it, and check the box that says remove source branch
- Send an email out to the team saying that there is a merge request to be reviewed and include the link
- Once the POC and at least one other person give it a thumbs up, the feature will be merged into develop

- The POC will make a merge request for develop into master and the PO will determine whether or not to accept the merge request

Items of Note:

- All pipelines `WILL` run the manage_enviornment, local build, unitests, static analysis
- Docker builds, management, registry pushes, and triggers will happen `NIGHTLY` on `MASTER AND DEVELOP ONLY!`
- It is the Responsibility of the POC to pull other people in on the merge if they feel like it is needed

Building the Source Code outside of the Pipeline

Build Source Code Locally

- From your home directory, cd into your directory where your source code is stored ie: cd git/dechannelizer
- Once you are in that directory run the following commands:
  - mkdir -p build
  - cd build
  - cmake3 ../
  - make -j4
  - make install (this will install to your home directory under the WEB folder)

Run Build Locally

- cd /home/`<User>`/WEB/bin/
- ./TransmitMultiplexer -c `<config_path>` --grpcport `<port_number>`
- If you are using the gRPC build and aren't debugging, you will want your config file to have:
  - "autoStart": false

Items of Note:

- `This Repo is dependent on API Services, Common, Process Manager, Chains, Dsp, Vrt, Ral and dspIO. you must build and install api_services, common, process_manager, chains, dsp, vrt, ral, and dspIo before you can run the cmake commands, otherwise the build will fail`

Build Docker Images Locally

- from your home directory, cd into your directory where your source code is stored ie: cd git/transmit-multiplexer
  - once you are in that directory run the following commands:
  - ./buildTransmitMultiplexerDockerImages.sh

Run Docker Image

`FOR NOW, WILL CHANGE`

Dechannelizer Docker Run Command

- docker run -ti -d --net=host --ipc=host --device /dev/nvidia0:/dev/nvidia0 --device /dev/nvidiactl:/dev/nvidiactl --device /dev/nvidia-uvm:/dev/nvidia-uvm --device /dev/nvidia-uvm-tools:/dev/nvidia-uvm-tools --name=`<Container Name>` `<Image Name>`
- `--entrypoint /bin/bash` can be added before the image_name to allow the container to start with just a shell.
- This command will create and run a container with an open shell, which can then be connected to in order to run Dechannelizer manually.

Run Dechannelizer Manually In Docker

- docker exec -ti `<Container Name>` /bin/bash
- cd /opt/ws/bin
- ./dechannelizer -c `<config_path>` --grpcport `<port_number>`
- The config you use may have the cuda device and pic device already defined.
- If you are using the gRPC build and aren't debugging, you will want your config file to have:
    - "autoStart": false