NA62 RunControl

Nicolas Lurkin

January 28, 2014

1 Finite State Machine

The NA62 RunControl is based on a three levels tree-like hierarchy of Finite State Machine (FSM).

Each constituting element of the DAQ (device) is internally modeled as an FSM. Each state of the FSM is defined by the evaluation of logical expressions depending on a set of parameters that are provided by the device. The figure 1a shows the FSM diagram followed by most of the devices.

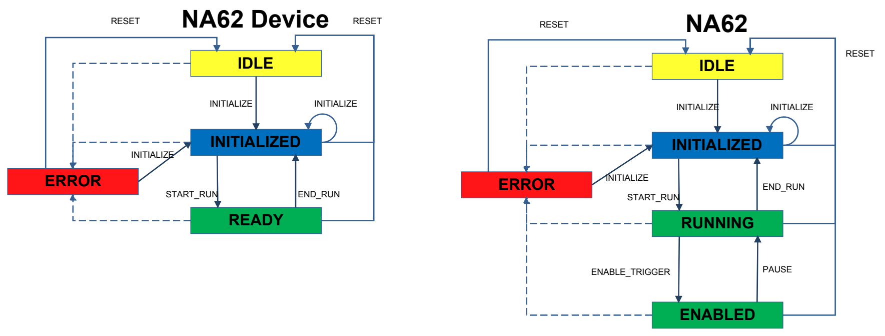
The device nodes are forming the leaves of the tree. The internal node are grouping the devices into logical entities representing subsytems of the experiment. These nodes are also modeled as FSM, summarizing the states of the devices belonging to this group according to a set of rules.

Finally the root of the tree is an FSM node that represents the global state of the Data Acquisition by further summarizing the states of all the logical nodes. The figure 1b shows the FSM diagram for this root node which is derived from the device FSM. The possible states are described below:

- **IDLE**: This is the initial state after starting or resetting the FSM and the devices.
- **INITIALIZED**: When all the devices have been configured and the DAQ is ready to take data.
- **READY**: All the devices are completely ready for data and waiting for triggers. The only exception being the trigger processor, in a paused state, waiting for further command to generate the triggers.
- **RUNNING**: The trigger processor is out of the paused state and running.
- **ERROR**: This state can be reached from any other state whenever a problem occurs on any device.

Each state allows a list of abstract commands that are propagated downward in the hierarchy to the device nodes where it is transmitted through the network. The list of commands is described hereafter:

- **INITIALIZE**: Request to initialize all the devices with a specific configuration.
- **STARTRUN**: Request to all the devices to start the run and move in a ready state where they are able to take data.
- **ENABLE_TRIGGER**: Request to the trigger processor to start generating triggers.



(a) Generic FSM diagram for the devices. (b) FSM diagram of the root node representing the global state of the NA62 DAQ.

Figure 1: Main Finite State Machine diagrams of the NA62 RunControl.

- **PAUSE**: Request to the trigger processor to stop generating triggers.
- **ENDRUN**: Request to all devices to stop taking data and end the current run.
- **RESET**: Immediately move to an idle/initialized state, stopping the current run if it was ongoing.

2 Interface

The link between the device nodes modelling a specific device and the hardware itself is established by DIM[1][1] that will take care of transmitting the commands from the RunControl to the device and the state parameters from the device to the RunControl, along with any other relevant information that should be known by the RunControl or made availabe to the shifters.

DIM is working on a client-server model where the same instance can be both server and client. The server part implements command and service ports. The clients can push commands/requests to the device via the first type. The minimum set of DIM commands to be implemented to interface with the RunControl is the following:

- dimServerName/Command: For the commands described in [2].
- dimServerName/FileContent: For the configuration files content according to the procedure described in [3].

While the service ports are providing information to the clients connected to the device. With these ports, the RunControl will be able to determine the exact state of the hardware, transmit information to the user and log them in the database. The minimum set of DIM services to be implemented is the following:

- dimServerName/Info: For transmitting output to the user. This service should be the equivalent of stdout when the control software is running in command line. It is only used to inform the user currently working on this specific equipment and will not be recorded.

[1]Distributed Information Management System
[2]TODO: ref
[3]TODO: ref

FSM State	Value
IDLE	0
INITIALIZED	1
READY	2
ERROR	Other (-99 reserved)

Table 1

- dimServerName/Logging: For the logging mechanism described in [4]. Contrary to the previous service, this one should only provide important and summarized information that will be stored in the offline database to be aware during future analysis of important problems that happened during the run.
- dimServerName/State: For FSM state repoting. If the device is internally keeping track of it's internal state, this state should be reported in this service. There should be a one to one correspondance with the FSM diagram 1a. The standard convention is shown in the table 1. The use of any other value for the ERROR state allows to define different type of errors identified by the error (state) code.

As the RunControl is not aware of the internal operation of any device the commands are very generic and the device is expected to understand them and execute the appropriate sequence of actions specific to itself. After the execution of the associated action the devices should answer back to the RunControl, notifying the success or failure of the action.

The commands are sent to the DIM xxx command port as a string. The first token of the string is the command line and the following tokens are the command arguments, if any. The minimum set of commands to be understood and implemented is the following:

- initialize
- startrun runNumber
- endrun
- resetstate

3 Configuration

The configuration mechanism of the RunControl will take advantage of the existing recipe mechanism of the JCOP[5] framework: a database contains an ensemble of fields and a list of recipes (configuration modes). Specific values of the field are associated to each recipe. At configuration time a prompt will ask the user to select a recipe to load.

Again, in order to hide the internals of the devices from the RunControl, the actual configuration parameters/values are contained in a configuration file. The content of the files will be written in the JCOP database and associated to recipes. When loading a recipe, the associated files content will be transmitted integrally to the device who will again be responsible to decode it and apply the values.

A tool will be available to dump configuration files into the database and an “on-line” editor will also be provided to modify/write files directly in the database. The entries in the database

[4]TODO: ref
[5]TODO: Link to JCOP

can be independant files or different versions of a same file and will be identified by a name and will be associated one or several tags relating it to recipes.

The configuration files can be incremental: for each recipe, a list of files can be sent to the device. The first one would contain “default” values, values that are valid for different type of runs, parameters that rarely change. The following files would contain parameters that have a higher changing rate. The files will be send to the device sequentially and the value applied for a specific parameter should always be the one specified in the last file (values in the latest file are overwriting the values in the earliest ones). The default file could also be used when no file is specified for the given recipe or when resetting the device.

The file transfer mechanism will work as described hereafter. After sending a command requiring a configuration file, the RunControl will send the content of the first configuration file as a string on a dedicated command slot. The RunControl will then wait until the device notifies it that the file has been entirely processed and is waiting for further instructions. The RunControl will repeat the same procedure for the next configuration files until the last one has been processed.

Once all the configuration has been applied the RunControl will ask the device to report back the current configuration. The device will generate a file containing all the current parameters values, possibly in the same format as the configuration file, that it will transmit to the RunControl. This file will be stored in the offline database. Two possibilities are given to the subsystem:

- When applying the configuration files, keeping track of the real value of each parameters (the one that has been applied) and report this list of values, trusting that everything went well and that these values were effectively loaded in the hardware.
- Request the hardware for the actual value of each parameter and report this list of true values.

When the complete procedure is finished the device should update it's state parameters and report an INITIALIZED state in the RunControl.

4 Logging

The RunControl will be connected to three different databases: the configuration database containing the recipes and the configuration files, the online database storing number of information related to the run and the instantaneous state of the data taking, PC farm, ... And finally the offline database that will contain the subset of the online values that are relevant to future analysis and the configuration for each run.

In addition to the database logging, there will be a centralized visual logging screen in the control room. Every device will implemment the xxx service port that implement the logging message. The message will contain the source id, a severity code and finally the text message itself.

5 Alternative/Additional functionalities

References

[1] http://dim.web.cern.ch/dim/