Table of Contents

RICH UKL1 Controls....1
Advanced installation....1
Using the FSM....1
Individual configuration....3
Recipe editor....4
Hardware editor....6
Errors and the exception log....6

RICH UKL1 Controls

More detailed guidelines on using the UKL1 PVSS controls project are provided here. These range from simple things like creating recipes and modifying the UKL1 registers directly to installing the UKL1 project from scratch. In the cases where an `x` appears in a name, for example `RxDAQL1`, then it can be replaced by a 1 or 2 depending on whether the RICH1 or RICH2 project is being dealt with. The previous example could become `R1DAQL1` or `R2DAQL1` and is the name of the RICH1 and 2 projects respectively.

Advanced installation

The components libraries `fwUkl1` and `fwUkl1ExceptionHandling` can be downloaded from the UKL1 archive and should be extracted into the `/group/rich/pvss/_lbRICHPackages/RxDAQL1/` directory. They can then be installed with the fwComponent installation tools as described on the RichOperations page.

In the event that the necessary PVSS project is not installed then a project should be created on the local disk of `rxdaq01` or `r2daq01`, for RICHX. The RICH1 project should be called and have system names of `R1DAQL1` and system number `136`; and RICH2 project `R2DAQL1` and `176`. The framework components should be installed in `/group/rich/pvss/fwComponents_RXDAQL1`. The framework components should be installed from `/sw/pvss/lhcbfw`, taking the highest version number unless an older version is required for some reason.

In certain cases it may be desirable to have the installation procedure completely recreate your FSM tree. In which case the list of UKL1s should first be deleted from the system via the `fwUkl1` panel or the maunal operations button in the FSM CU control. Then the configurator object manually removed from the FSM tree, finally the CU can be removed. The is done through the device editor and navigator, DEN panel. The FSM tab is selected and set to editor mode, there right clicking on the RICHX CU will allow you to access the FSM configurator options, which provides a delete option. The RICHX CU can then be remove through the right click menu. The panel `fwUkl1` can then be used to recreate the tree. It provides a button `Find UKL1 boards`, which is used to recreate the FSM tree. The `fwUkl1` panel is installed as a manager in the PVSS console and is provided as a short cut (listed details on RichOperations) in the LHCb online environment. With the FSM tree deleted the UKL1 CU panel can no longer be accessed!

Using the FSM

Double clicking on the shortcut will open a panel that allows the control of the UKL1 FSM tree. You will be presented with a tree structure with the top level node called `RICHx_L1`, which contains sub-nodes with the following naming structure `rxukl1yy`, where yy is a decimal number e.g. `r2ukl104` and represents the production number of the board.

A brief diversion to explain this structure will be given introducing some of the FSM terminology. The FSM, finite state machine, represents defines a finite number of states that the UKL1 boards can be in and within each state there are a selection of actions that can be performed. The actions vary depending on the state and it is by performing these actions that state transactions are triggered. The top level node `RICHx_L1` is a control unit, CU, which can be used to send commands to all of the individual UKL1 boards. Operations issued from this level will affect all enabled UKL1 boards in the FSM and a panel is provided for the UKL1 CU level which provides the ability to control and operate a collection of UKL1s. Each UKL1 board is a device unit, DU, in the tree and represents a physical UKL1 board. Here actions that are performed affect only the select UKL1 and custom panels are performed that allow the status of the individual UKL1 boards to be monitored and the manipulation of settings at the register level to be performed.

In order to control the UKL1 boards and to perform actions the UKL1 CU, `RICHx_L1`, should be first selected with a single left click and then right click and select `View` from the presented menu. This should open the

panel for controlling the UKL1 boards at the CU level. Along the top of this panel should be the `System` and `State` sections, with the system set to `RICHx_L1` and the state could be any valid UKL1 state. Down the left hand side of the panel there will be a list of `Sub-systems` and their `State`, with each subsystem representing a physical UKL1 board. Also present should be the FSMConfDB object, called `RICHx_L1_ConfDB`, which manages the recipes and the database interactiond. This object, known as the `Configurator` should be present but its use should be completely transparent to the User. It will respond as a DU to CU commands and should transition between the states as any UKL1 board, more information about this object can be found here.

In order to trigger a state transition click on the state and select the desired action either of the CU to trigger the action for all UKL1 boards and configurator or individual DUs to affect only that UKL1. This will trigger the state change, which in most cases will be successful and the new state will be entered. On occasion an error will occur that will cause the boards to enter either the `ERROR` or `NOT_READY` states, errors will be discussed in a later section.

The DU panels can either be access by selecting them in the FSM tree, right clicking and selecting `View` as for the CU or by double clicking on the relevant sub-system name on the CU FSM panel. From here various settings can be modified or viewed for the individual UKL1. The configurator object can be accessed in the same way, but there is no functionality that can be accessed via this panel.

The following actions are available and their parameters described.

`Reset`
: Reloads the firmware on the UKL1 boards and puts them back into an unconfigured working state. Affects the boards in all states except the `RUNNING` state.

`Recover`
: The same action as reset, but only affects boards in an `ERROR` or `UNKNOWN` state.

`Configure`
: Loads a recipe into the UKL1 board. A relatively slow step, it take ~1min to configure 8 boards. In order for the boards to configure they must be given the following parameters:

  `PART_ID`
  : Partition ID that the UKL1 is in. This is calculated dynamically if configured from the ECS level and if not it generally can be ignored and it will be configured from the recipe.

  `RUN_TYPE`
  : This is the name of the recipe to load. The defined recipes for the UKL1 are listed below. If this is not set then the `DEFAULT` recipe will be loaded.

  `N_STEPS`
  : This is ignored by the UKL1 boards. It defaults to zero, but will be set during calibration runs where the TFC will automatically step through a range of settings. There is no present need for the UKL1 to respond to this parameter.

  `RUN_INFO_DP`
  : This is the name of the run information DP that contains information about the run conditions. It is accessible from the ECS level under the run info subsystem. It is from this the UKL1 will take the MTU size, which defines the maximum size of the ethernet frame containing a MEP. This can be set from the sub-detectors section of the run information sub-system. If not running with the top level ECS then this setting can be left empty and the value is taken from the recipe.

`Start`
: Enables the UKL1s to receive and respond to triggers. Takes the `RUN_NUMBER`, which is ignored by the UKL1. The ODIN bank will contain this information.

`Stop`
: Prevents the UKL1s from responding to triggers. No parameters are required for the `Stop` action and the UKL1 should be in the `READY` state once completed.

`FAKE_READY`
: This action can only be accessed from the `CONFIGURING` state and is used in the event that there is a problem with configuring that prevents the state transition completing and it does not time out correctly. It will force the board into the `READY` state. It cannot be used to interupt the configuring

process as it does not interupt the current action and proceeds after it has completed.

`AbortCommand`
: This interupts the current command sequence and forces the state to be reset to `NOT_READY` as it interupts the current action a `Reset` is recommended after this action.

The following recipes are defined for use with the `Configure` action:

`PHYICS`, `LHCB`, `LHCb` and `DEFAULT`
: These all configure the UKL1 boards into the same state which is to receive LHCb triggers.

`ALICE` and `CALIB`
: These all configure the UKL1 boards into the same state which is to receive ALICE triggers.

The list of states defined for the UKL1s and their meaning are given below.

`NOT_READY`
: This is the up state of the UKL1s and the also the state that is entered after recovering from an error or following a reset or reload. It is the state of the UKL1s when they have not been configured or the settings in the UKL1 registers is no longer know to the FSM. This state can be spontaneously entered if the connetion to the `ccserv`, which manages the interaction between PVSS and the hardware registers, is lost. This will most likely be caused by a network error as the `ccserv` runs on the CCPC on the UKL1. The UKL1s will return to the `NOT_READY` state when the connection is re-established. If the UKL1 board is in the `RUNNING` state this will not happen and an error will be sent to the exception log instead.

`CONFIGURING`
: When in this state it indicates that the UKL1 board is being configured and the registers are being configured from a predefined recipe. This is a transition state and the UKL1s should automatically leave this state once they have been configured.

`READY`
: In this state the UKL1s having been configured for a specific run, are ready to take data but will ignore TFC commands, such as triggers.

`RUNNING`
: In this state the UKL1 has been configured and is now ready to take data, it is also listening for triggers and other TFCs commnads. In this state data can be seen by the UKL1, it will be processed and if told to send the data to the event filter farm.

`ERROR`
: This state is entered after any action fails to complete successfully. More details about why this state was entered can typically be found in the exception log.

`UNKNOWN`
: This is the `else' state for the UKL1 boards. It occurs rarely and indicates that the UKL1 boards are in an undefined state. This will never indicate an actual problem with the UKL1 boards and they can be triggered back to the `NOT_READY` state. If the boards have entered it is likely they have not been initialised correctly and errors should be checked for. The developer should be contacted with detail about how and where this state was encounter. Errors should also be checked for as they may provide some clues as to what happened.

Individual configuration

By accessing the DU panel as described above a panel for accessing the individual UKL1 settings is provided. Two buttons, titled `Status` and `Configuration`, which when clicked produces a drop down box with a list of options allows the User to monitor the status of the UKL1, create and manipulate recipes for the specific UKL1 board and also to access the hardware registers for manual configuration. A brief explanation of how to use the status panels is given here and then the hardware and recipe editors are explained in the following section.

The UKL1 board has been divide into three sections based on the hardware design. There is the Ingress FPGAs, of which there are four, that receive data from groups nine channels and perform error checking on the events and format the channel data for inclusion in the event. The Egress FPGA collates the events from each of the Ingress FPGAs and compiles them into individual events, creates multi-event packets, MEPs, from these events and finally fragments the MEPs for transmission over ethernet and sends them to the gigabit ethernet, GBE, card. The GBE card is the finally group that the software implements and this is responsible for sending the MEPs via the ethernet link where it will eventually reach the event filter farm. It is these three groups Ingress FPGA, Egress FPGA and GBE that are consistently refered to in the software and are used to group related settings and status information.

Status panels are provided for Ingress and Egress FPGAs, and GBE card and can be accessed from the `Status` drop down menu (explained above), also an Overview panel is provided to display status values that are relevant to the UKL1 board as a whole. The values on the panels will be updated from the hardware where when the panels are first loaded, but in order to monitor them for changes the `Start monitoring` button must be click. If they are to be no longer monitored then a `Stop monitoring` button exists to stop monitoring. It is not possible to know whether registers are currently being monitored and thus to ensure monitoring is started or stopped the relevant button should be clicked. It does not hurt to start monitoring when it is already started and to stop it if it is already stopped. Further the monitoring can be configured via the `Config monitoring` button, the update frequency of the displayed registers can be altered or they can be set to be updated only when a value changes, this cuts down on network traffic between UKL1 boards and the PVSS project. The default state for monitored registers is every 2s on data change. The GBE is a slight exception to this as it monitors automatically when the panel starts and also provides the ability to manually enable or disable the ports as well as indicating the status.

Information about the displayed values can typically be found by hovering over the text field. Any values that cannot be displayed due to an error will be highlights in yellow and will display the text `Fail`. Values that indicate the UKL1 is in an error state will be highlighted in orange. Any other colours (should only be white!) indicate there is no problem with the board being in this state, they could of course indicate why a particular effect is being seen as it may not be what the UKL1 board considers as an error. Certain values are only valid in the case of certain status bits being good, in this case these will be flagged as `N/A` and if they high some kind of error with the UKL1 board they will be orange, otherwise if it is because a non critical bit is disabled they will be white.

Recipe editor

Recipes all predefined settings to be loaded into the registers of a UKL1 board triggered by certain actions. All recipes have a recipe type associated with them. The recipe type defines the registers that are stored in a recipe and the recipe contains the values for those registers. The UKL1 project has defined two recipe types:

`UKL1_Configure`
: Contains all the registers that are required by the `Configure` action and are loaded during the `CONFIGURING` state. It contains settings for all three sections, Ingress, Egress and GBE, and thus all must be configured through the recipe editor. All recipes of this type should contain for the first part of their name the run mode e.g. PHYSICS, LHCb, ALICE, CALIBRATION etc. and are typically in capitals although there is no standard for this. The second part of the name is `/Configure`, case sensitive, and without this the recipe cannot be found by the FSM.

`UKL1_Start`
: This contains the settings that must be set when performing the `Start` and `Stop` actions to enter the `RUNNING` and `READY` states respectively. It contains only two settings at present, TfcDecoder and GbePolling, that must be both set to either `Enable` or `Disable` for use with the `Start` and `Stop` action respectively. As mentioned above all UKL1s must have a `PHYSICS/Start` and `PHYSICS/Stop` recipe defined which are of this recipe type with the settings just described.

The recipe editor is selected from the menu list on the DU panel and here a list of the available recipes is seen along with the ability to create new recipes, edit and delete existing recipes.

Creating recipes

The recipe type must be choosen and then the name of the recipe given. The format of the name is very important and must follow the rules above for `UKL1_Configure` type recipes and that there is currently no use for recipes of `UKL1_Start` type that are not called `PHYSICS/[Start,Stop]`. All newly created recipes will have prepended to them the name of the DU and this need not be added by the User.

- Errors: The most like source of error when creating a recipe is that it has been seen that it can take 2-3s to load the available recipe types in the drop down list. A clicking happy person might therefore be able to click `OK` before the recipe types are loaded and thus selected an empty string for the type. This will abort recipe creation with a scary error message prompting the User to contact an expert if no types were available. It is possible that the recipe types are just taking an excessive time to be retrieved and loaded (2-3s has been seen in the SSB2 lab), just wait a little longer to see if the list becomes populated. If it does not then consult an expert as recipe types cannot be created via the UKL1 project.

Editing recipes

First a recipe must be selected, via a left click, from the list of available recipes and then the `Edit recipe` button should be clicked. From here the recipe settings for the three sections can be selected from. Again the recipe type must be selected, this should be the same as for the last time the recipe was saved! For `UKL1_Configure` type recipes settings for all the sections must be set. At present in the pit only port 0 is connected on the GBE and the recipe must be set to reflect this. There are a few other settings that must be set correctly, which are given in a moment, the rest can be set to the hardware defaults typically and an explanation of how to find these is in the hardware editor section. The value for the latency, which is set on the Egress FPGA, is given below, the source MAC and IP addresses are given on the TELL1IPRanges page, the destination address is also given here (destination IP address can be seen in the sample TELL1 .cfg file). Note that the last two bytes of the UKL1 source MAC address and last byte of the source IP address are configured automatically and not set in the recipe editor (it will prevent you from doing so). The UKL1 firmware can autodetect which channels are active and so the recipe does not need to ensure it configures the UKL1s for this. Generally the L0 emulator in the Ingress FPGA settings should be disabled in any recipe unless you wish the UKL1s to autogenerate events! A final note on editing recipes. When saving values to a recipe the `Apply` button should first be clicked, `Close` will merely close the panel and not save any settings. This means that close can be used as a cancel i.e. exiting without changing anything if a mistake is made and the recipe does not need to be altered.

- Errors: If a newly created recipe is loaded it typically will not have any setting associated with it (it can under specific circumstances). Currently this will generate an error when initialising the editing panel and the exception log will complain that it could not find enough data to convert. This is normal and an irritation in the current version. Simply save the recipe and the error indicator on the panel should be cleared. Errors are show in the exception log and the most common is for an incorrectly formatted entry. It should be detailed in the log and can be corrected. If recipe values do not appear to be being saved and there are no messages in the exception log then it is most likely that `Apply` was not clicked before `Close` in the recipe editor panel. As mentioned in the creating recipe section if the recipe types are slow to load it is possible to continue without selecting an appropriate type if return is clicked too fast. This will abort the recipe editing, again try wait 2-3s to see if they appear, otherwise consult an expert. For other errors with no obvious exception log entry an expert should be consulted.

Deleting recipes

Select a recipe by left clicking and then click the `Delete recipe` button. This will delete the selected recipe and are there is no delete confirmation!

- Errors: Delete can pretty much only fail if a recipe is selected for deletion that does not exist. Try clicking `Refresh` to ensure that the selected recipe has not already been deleted. Otherwise consult an expert.

Hardware editor

The hardware can be edited via DU panel options list again only for one of the three sections at a time. These are marked by Ingress or Egress config, as mentioned above the GBE is the exception to this and contains both the ability to edit the settings and view the status of the GBE. This is due to the fact it contains so little editable settings. The Ingress and Egress config panels display the current values in the various configurable registers for these sections, which can be monitored in the same way as for the status panels. In order to edit these settings the button `Hardware config` should be selected. This will open a panel that is the same in appearance to the recipe editor for the corresponding settings and the values are set in exactly the same way. Click `Apply` to save any settings and `Close` will exit, but does not save. The panel is populated with the current hardware values when it loaded and when the values are written they will automatically update on the config status page.

Errors that occur while writing to the exception log will be sent to the exception log and will likely occur due to either poorly formatted values for the specified register or problems writing to the hardware. Poorly formatted data should be indicated in the log and should be easily solved. Problems with the hardware connection are more complex and will likely require expert intervention.

Errors and the exception log

The UKL1 project follows the framework guidelines for passing and handling exception information for function calls. Each exception has a severity level, code and message. The framework defines three severity levels warning, error and fatal and two exception codes have been defined for use with the project 1 and 2.

Exception code 1
: Indicates that a function failed to complete successfully and return prematurely. Can be coupled with the fatal and error severity levels. If coupled with fatal then the UKL1 will be in a state that requires major intervention to recover from. Coupled with error and the UKL1 is experiencing problems but it is likely to be relatively easy to fix, it will have left the UKL1s in an unusable state.

Exception code 2
: Indicates that a function completed successfully but for various reasons may have failed in 1 or more tasks that it was required to do. Can be coupled with error and warning levels. If coupled with an error severity level then this indicates there was a failure somewhere during the execution, but the board will still be usable if that setting that failed to be applied isn't vital to the current run. When issued with a warning then the function failed to complete a task, but was able to massage the failure somewhat and make it successful. This may or may not have corrupted the UKL1 running state depending on the requirements.

The display of these errors is done via custom UKL1 libraries that display the exception messages from each function in a tree structure. The top node in the tree will represent the hardware that caused the error and each sub node of the tree represents a new exception chain and labelled as a unique number. Each exception chain is generated for a specific function call, with this function call is displayed as the first node in the tree. Any subsequent functions that were called by this function that failed will appear as nodes in the tree, forming a new level each time. If a given function failed multiple times within a single function call they will all appear at the same level. Thus a detailed tree of the function call chain can be built up, each of which will have an error message indicating what went wrong.

The first function to be displayed is the 'highest level' function that will contain a fairly User friendly error message and should provide a basic idea of what was failed to be achieved. By navigating down the exception tree more detailed error messages will be encountered that provide more detailed information about what caused the failure and will often only be interetable by a UKL1 project expert. This is still in the early stages of development and the author would appreciate if any oddities in the tree display or grammatically dubious error messages were reported to Gareth Rogers.

As the UKL1 project relies on many other projects, not all of which use the same error reporting mechanism, often further details about the errors can be found in the PVSS logviewer and any information would likely be of use when reporting problems to experts.

At present there is a bug that results in the `Verify write` option on the hardware editor panel failing, with apparent inconsistencies between the written hardware values and those read back. This is a known bug and the hardware config status panel should be consulted to ensure the settings are as requested. If the settings are not as requested then a more serious error has occurred. This option should be avoided in version 1.3 (current IP8 installation) and below of the UKL1 projects.

If problems are seen with configuring due to difficulties with the TTC or the serial number and UKL1 temperatures (Overview in DU panel) show rubbish or fail to be acquired then it is possible that the UKL1 has incorrectly been identified as a prototype board. This can happen if the FSM tree is created too soon after the project is started and the UKL1 boards have not yet been properly configured for communication. The problem is easily fixed, but first the relevant pc (`r[1,2]daq01`) should be logged into and the PVSS console start, `startConsole` in the terminal. In the console a `PVSS00ui` manager with the options `-p fwUkl1/fwUkl1.pnl -iconBar -menuBar` should be started. This is the UKL1 project administrator panel and here there is a button labeled `Reload list`, click this and it will recreate the list of UKL1 boards in the FSM tree. Check that no errors occurred as a warning is issued if the boards were identified as a prototype and then check the serial number and temperatures again. Once this has been done the project can be run from the FSM tree display that is started via the `R2DAQ01_UI_FSM` shortcut mentioned previously.

If the UKL1 is repeatedly identified as a prototype it is likely that it no longer exists in the system. It takes up to 5 minutes for a board to be registers as missing if its connection is terminated abruptly and uncleanly. Further symptoms that a board is missing from the system is a failure to read and write to any of the registers on the board. If after 5 minutes or so the UKL1 board that is causing the problem ceases to exist in the list then it has left the system. It is likely that this was an unexpected event as the FSM tree would have been recreated successfully in the event of a known removal of the board. An expert should be consulted on this matter.

If all the boards fail to respond to reads, writes, successful recreation of the list then the is possibly something wrong with the DIM server. In order to check this the relevant pc must be logged into (`r[1,2]daq01`) and the PVSS console started (explained here). In the console there is a `PVSS00dim` manager that should be green, if this is flashing blue and yellow there is definately a problem with the DIM server (responsible for communication between the UKL1s and PVSS). In the case the `DIM_DNS_NODE` environment variable should be checked to ensure it is set correctly, to `r1daq01` or `r2daq01` i.e. the pc running the PVSS project. If you don't know how to do this already then you probably shouldn't. If it is set correctly or the PVSS DIM manager is green then it is possible that the crate has been turned off and all the boards have abruptly left the system. If after 5 minutes no boards can be found an expert should definitely be consulted.

A few possible/common/easy to solve errors are outlined here and many more are possible (and many not yet encountered). Your nearest UKL1/RICH expert (UKL1 expert preferable) should be consulted when any strange or unexpected behaviour is encountered.

-- GarethRogers - 02 Dec 2007

Topic revision: r2 - 2008-05-06 - GarethRogers