C. Perkins
210929

DSM2 / QTD / STP2 Configuration File Manual

The top-level configuration file is located on startrg in `~trg/cfg/STP2/`. This configuration file will be used to configure all DSM2, QTD, and STP2 boards in the system and contains the register settings for all of these boards. Therefore, this configuration file must be updated anytime one of these boards is added or removed from the system, whenever a register needs to be added or removed from the Run Control GUI, or whenever a default register setting needs to be changed. This configuration file should be updated under similar circumstances as when the Tier1 file is updated except that this file corresponds to DSM2, QTD, and STP2 boards whereas the Tier1 file corresponds to DSM1, QTB, QTC, and RCC2 boards.

The connection to the Run Control GUI requires that the top-level configuration filename be `trg_dsm2_qtd_stp2_config.version.txt` where `version` is replaced with a ***number*** used to identify the file. The Run Control GUI requires that the version be a number. The convention is to use a version number that contains the date and a revision number : YYMMDDRR (YY = year, MM = month, DD = day, RR = revision).

For example, here is the config file : `trg_dsm2_qtd_stp2_config.21082500.txt` :

```
# TRG_GROUP    Name          STP2_Config_File                          RC_Port  RC_Node_ID

TRG_GROUP    STP2          STP2/trg_stp2_config.21082500.txt         4000     0x801b
TRG_GROUP    L1_DSM2       L1_DSM2/trg_l1_dsm2_config.21082500.txt   4000     0x801c
TRG_GROUP    BBC_DSM2      BBC_DSM2/trg_bbc_dsm2_config.21082500.txt 4000     0x801d
TRG_GROUP    BBQ_QTD       BBQ_QTD/trg_bbq_qtd_config.21082500.txt   4000     0x801e
TRG_GROUP    MIX_DSM2      MIX_DSM2/trg_mix_dsm2_config.21082500.txt 4000     0x801f
TRG_GROUP    MXQ_QTD       MXQ_QTD/trg_mxq_qtd_config.21082500.txt   4000     0x8020
TRG_GROUP    BCE_DSM2      BCE_DSM2/trg_bce_dsm2_config.21082500.txt 4000     0x8021
TRG_GROUP    BCW_DSM2      BCW_DSM2/trg_bcw_dsm2_config.21082500.txt 4000     0x8022
TRG_GROUP    BC1_DSM2      BC1_DSM2/trg_bc1_dsm2_config.21082500.txt 4000     0x8023
TRG_GROUP    EQ1_QTD       EQ1_QTD/trg_eq1_qtd_config.21082500.txt   4000     0x8024
TRG_GROUP    EQ2_QTD       EQ2_QTD/trg_eq2_qtd_config.21082500.txt   4000     0x8025
TRG_GROUP    EQ3_QTD       EQ3_QTD/trg_eq3_qtd_config.21082500.txt   4000     0x8026
```

Each line in the top-level config file corresponds to a node in Run Control and there should be one line/node for each crate in the system plus one additional line for the `STP2` node. Anything after a `#` is considered a comment and is ignored. Any empty lines in the file are skipped.

The first field on a line is a keyword. Currently, the only supported keyword is `TRG_GROUP` which specifies a new group of trigger boards (i.e. a new node). The group of boards can contain any combination of DSM2, QTD, or STP2 boards and the boards don't have to be physically in the same crate, although currently the boards are grouped by crate.

The second field is the name of the node. The third field is the configuration file for the node using a path relative to the top-level configuration file. The fourth field is the port to be used to connect to the Run Control handler. Currently all of our nodes connect to port 4000. The fifth

field is the Node ID as used in Run Control and must be the Node ID expected for that node as assigned by the DAQ group.

Each of the groups' configuration files follow a similar format to the top-level file. Anything after a # is considered a comment and is ignored. Any empty lines in the file are skipped.

Within the group configuration file, the section for each board begins by defining the hostname of the ConnectCore module on the board. Any lines after the hostname line apply to that board until the next hostname line is encountered. The configurations of the different types of boards (i.e. DSM2, QTD, STP2) follow similar formats but have some different keywords available which will be described individually below.

STP2C Board Configuration Keywords :

Example board configuration section for STP2C :

```
# stp2c-bbc
STP2C_HOSTNAME    stp2c-bbc
STP2C_DISABLE     0
STP2C_NODE_ID     0x8206
STP2C_REG         0x004    0x8206   STP2C_BBC_NODE_ID                 # Node ID
STP2C_REG         0x021    0x800d   STP2C_BBC_STP1_NODE_ID            # STP1 Node ID
STP2C_REG         0x019    0xffff   STP2C_BBC_TX_PORT_EN              # STP2 TX Port Enables
STP2C_REG         0x01f    0xffff   STP2C_BBC_RX_PORT_EN              # STP2 RX Port Enables
STP2C_REG         0x020    0x0101   STP2C_BBC_STP1_PORT_EN            # STP1 TX/RX Port En
STP2C_REG         0x071    0x0001   STP2C_BBC_STP1_RX_FAST            # STP1 RX Fast/Slow
STP2C_REG         0x072    0x0000   STP2C_BBC_STP1_TX_FAST            # STP1 TX Fast/Slow
STP2C_TX_TYPE     0xffff   0x0000   STP2C_BBC_TX_TYPE_1       # Disable All Packet Types on All Ports
STP2C_TX_TYPE     0xfffe   0x0009   STP2C_BBC_TX_TYPE_2       # Enable Build_Event and Test Packets on Ports 2-16
STP2C_TX_TYPE     0x0001   0x0004   STP2C_BBC_TX_TYPE_3       # Enable Data Packets on Port 1
```

Keyword definitions :

`STP2C_HOSTNAME`:

Specifies the hostname of the ConnectCore module for the STP2C board and starts a new section for a board. All lines that follow this one apply to this board until the next hostname keyword is encountered.

`STP2C_DISABLE`:

Specifies that the board should be ignored (i.e. disabled) but allows the configuration information to remain in the file so that it can be quickly re-enabled if necessary.

`STP2C_NODE_ID`:

Specifies the Node ID for the board. The list of Node IDs for all boards is currently located on l2ana01 in `~trg/online_l2/trgNodes.h`. This Node ID is used to identify nodes in L2, packets within the STP2 network, and the data packets coming into L2. All nodes should have a unique Node ID even if they do not send data to L2. The Node IDs for trigger boards are all internal to the trigger system and are assigned by the trigger group.

STP2C_REG：

Example :

```
STP2C_REG   0x019   0xffff STP2C_BBC_TX_PORT_EN  BBC_STP2C_TX_PORT_EN  # STP2 TX Port
Enables
```

Specifies a register to be written to the board during configuration.
The second field for this keyword is the register address which can be in either hexadecimal or decimal format. The third field is the register value which can also be in either hexadecimal or decimal format.
The fourth field is a name for the register. This name is for archival purposes within trigger and should be unique for each register even if multiple registers share the same purpose.
The fifth field is a dictionary name for the register. This is what will show up in the Run Control GUI and follows the normal rules for Run Control dictionary names (i.e. registers in the GUI will be grouped according to the characters before the first "_"). If registers share a common purpose, they can use the same dictionary name and will only show up once in the Run Control GUI but will be assigned to all registers with that name (i.e. same as currently done in the trigger Tier files). If the register should be written at configuration but NOT show up in the Run Control GUI, this dictionary name field can be omitted.

Each register will be written at run configuration time (either the value in the config file or any over-ridden values in the GUI) and then readback to verify that it was written correctly. If a register doesn't read back correctly, configuration will fail.

STP2C_REG_NR：

This also specifies a register to be written to the board during configuration (see STP2C_REG keyword) but does NOT require correct readback to succeed. Some registers need to be written but the readback value is not the same as what is written and this should not cause the configuration to fail. The format of this line is the same as STP2C_REG.

STP2C_TX_TYPE：

Specifies the types of packets that should be transmitted on each port of the STP2 board. These lines should not need to be changed after their initial setup.
The second field specifies which ports should be configured by this line. Each STP2C board has 16 full duplex ports so this field is a 16-bit value with one bit corresponding to each port. The least significant bit corresponds to the first port on the board.
The third field specifies what types of data should be enabled for the ports specified in the previous field. Each bit in the value corresponds to a type of data packet, defined as follows :

```
Bit 0 - Build Event Packet
Bit 1 - Token Release Packet
Bit 2 - Data Packet
Bit 3 - Test Data Packet
```

STP2R Board Configuration Keywords :

The STP2R keywords are the same as the STP2C keywords but use `STP2R_` instead of `STP2C_`.

DSM2 Board Configuration Keywords :

Example board configuration section for DSM2 :

```
# dsm2-1 - Slot 6 - BB101 - Data Slot 0
DSM2_HOSTNAME              dsm2-1
DSM2_DISABLE               0
DSM2_NODE_ID               0x8211
DSM2_DATA_CRATE            8
DSM2_DATA_SLOT             0
DSM2_DATA_REQUIRE          0
DSM2_DATA_CREATE_STRUCT    11
DSM2_REG          0x000004     0x8211        DSM2_1_NODE_ID    # Node ID
DSM2_REG_NR       0xfffff0     0xabc         DSM2_1_TEST_REG1          BBC_TEST_DICT_REG1  # DSM2 Test Reg 1
DSM2_REG_NR       0xfffff1     0xabd         DSM2_1_TEST_REG2          BBC_TEST_DICT_REG2  # DSM2 Test Reg 2
```

Keyword definitions :

`DSM2_HOSTNAME`:

Specifies the hostname of the ConnectCore module for the DSM2 board and starts a new section for a board. All lines that follow this one apply to this board until the next hostname keyword is encountered.

`DSM2_DISABLE`:

Specifies that the board should be ignored (i.e. disabled) but allows the configuration information to remain in the file so that it can be quickly re-enabled if necessary.
‘1’ = Disabled, ‘0’ = Enabled

`DSM2_NODE_ID`:

Specifies the Node ID for the board. This Node ID is used to identify nodes in L2, packet sources within the STP2 network, and the data packets coming into L2. All nodes should have a unique Node ID even if they do not send data to L2. The Node IDs for trigger boards are all internal to the trigger system and are assigned by the trigger group. The list of Node IDs for all boards is currently located on l2ana01 in `~trg/online_l2/trgNodes.h`

`DSM2_DATA_CRATE`:

Specifies the `CONF_NUM` for the crate that the board is in. When a DSM2 data packet arrives in L2, it will be copied into the data block corresponding to this parameter. For example, if this parameter is “8”, the DSM2 data will be copied into the BBC crate’s data block. The `CONF_NUM` definitions are located on startrg in `~trg/trg_soft_dev/include/trgConfNum.h`

`DSM2_DATA_SLOT`:

Specifies the position in the data block that the DSM2 data packet should overwrite. Once the data block is determined using the `DSM2_DATA_CRATE` specified above, the position *within* the block is determined using this parameter. A value of "0" specifies that the DSM2 data should overwrite the first DSM's data in the block, a value of "1" overwrites the second DSM's data in the block, etc.

`DSM2_DATA_STRUCT_CREATE`:

Specifies that when this board's DSM2 data packet arrives in L2, the data block for that crate should be created within the trigger data structure. The second field in this line specifies the number of DSM data slots that should be created in the data block. Traditionally, this data block was created by L2 when the data packet arrived from the VxWorks node but in the case that a node contains only DSM2 boards, the VxWorks node will send an empty packet just so that it doesn't timeout the event. L2 then instead creates the data block when the specified DSM2 board's data arrives. Therefore, this parameter should only be defined for one board in a crate and only for crates with all DSM2 boards. This parameter can be set for any one of the DSM2 boards in the crate but the convention will be to set it for the first board in the crate.

`DSM2_DATA_REQUIRE`:

Specifies whether or not this board's data packet is required to arrive in L2. If the board's data is required in L2 (value = 1) but doesn't show up for an event, the event will timeout. If the board's data is *not* required in L2 (value = 0), it is fine to send it anyway but the event won't timeout if it doesn't arrive and L2 won't wait for the packet to arrive to process the event.

`DSM2_REG`:

Specifies a register to be written to the board during configuration.
The second field for this keyword is the register address which can be in either hexadecimal or decimal format. The third field is the register value which can also be in either hexadecimal or decimal format.
The fourth field is a name for the register. This name is for archival purposes within trigger and should be unique for each register even if multiple registers share the same purpose.
The fifth field is a dictionary name for the register. This is what will show up in the Run Control GUI and follows the normal rules for Run Control dictionary names (i.e. registers in the GUI will be grouped according to the characters before the first "_"). If registers share a common purpose, they can use the same dictionary name and will only show up once in the Run Control GUI but will be assigned to all registers with that name (i.e. same as currently done in the trigger Tier files). If the register should be written at configuration but NOT show up in the Run Control GUI, this dictionary name field can be omitted.

Each register will be written at run configuration time (either the value in the config file or any over-ridden values in the GUI) and then readback to verify that it was written correctly. If a register doesn't read back correctly, configuration will fail.

`DSM2_REG_NR`:

This also specifies a register to be written to the board during configuration (see `DSM2_REG` keyword) but does NOT require correct readback to succeed. Some registers need to be written but the readback value is not the same as what is written and this should not cause the configuration to fail. The format of this line is the same as `DSM2_REG`.

QTD Board Configuration Keywords :

The QTD keywords are mostly the same as the DSM2 keywords but use `QT32D_` instead of `DSM2_`.

Default Startup Configuration :

When a node starts up, it automatically loads the configuration parameters for all boards in the system, as specified in the top-level configuration file :
`trg_dsm2_qtd_stp2_config.default.txt`

If new default startup parameters are required, simply change the parameters in this configuration file. A new dictionary file does *not* need to be created for these default parameters - the nodes read the configuration file directly.

Creating Dictionary Files from Configuration Files :

When a new DSM2 / QTD / STP2 configuration file is made, a corresponding dictionary file must also be created so that dictionary entries show up in the Run Control GUI. To create the dictionary file from a configuration file, log into either startrg or scaler48 and run :

```
trg_cfg_make_dictionary  <version>
```

For example :

```
trg_cfg_make_dictionary  21082500
```

This will create the dictionary file `trg_dsm2_qtd_stp2_dict.21082500.txt` from the top-level configuration file `trg_dsm2_qtd_stp2_config.21082500.txt` located in the standard DSM2/QTD/STP2 config file directory. The standard filename prefix of `trg_dsm2_qtd_stp2_config` is always used for the config file and the standard filename prefix of `trg_dsm2_qtd_stp2_dict` is always used for the dictionary file.

After a dictionary file is created, it is automatically scp'd to the correct location on startrg to be picked up by Run Control.

Note that a new dictionary file does *not* need to be created each time a new Tier1 file is created. The dictionary file only needs to be created when the DSM2/QTD/STP2 configuration file changes and the resulting dictionary file can be re-used in multiple Tier1 files.

To include a DSM2/QTD/STP2 dictionary file in a Tier1 file, simply include the DSM2/QTD/STP2 ***dictionary*** file in the Tier1 .wild file ***before*** making the Tier1 file (i.e. MakeConfig) :

```
# DSM2/QTD Cfg File Version and Registers
include trg_dsm2_qtd_stp2_dict.21082500.txt
#
```

Interface with Run Control GUI :

The above procedure explains how to create the dictionary file from a top-level DSM2/QTD/STP2 configuration file (and include that dictionary file in the Tier1 file) so that the desired dictionary registers show up in the Run Control GUI.

In addition to all the GUI registers, one line is automatically added to the DSM2/QTD/STP2 dictionary file by the `trg_cfg_make_dictionary` program, for example :

```
40 11 1 TIER1_DSM2_QTD_CFG_VERSION 21082500
```

This line allows all the nodes to know what DSM2/QTD/STP2 configuration file is being used by a given Tier1 file. So for example, when a run is configured, each node finds this line in the GUI Labels, builds the configuration filename from the version number, configures its boards' registers' default values according to this configuration, and then over-rides with any registers set in the GUI.

Using this interface, different types of run configurations (i.e. data-taking, pedestal, cosmic ray, etc) all use the standard configuration filename of `trg_dsm2_qtd_stp2_config.version.txt`. The "revision" in the version number can be used to distinguish between different configurations created on the same day and therefore between different types of run configurations.

If it becomes difficult to identify different types of run configurations simply from the version number, a soft-link could be used to a more descriptive filename (i.e. `ped_dsm2_qtd_stp2_config.21082500.txt`, `cr_dsm2_qtd_stp2_config.21082500.txt`, etc) but it is still the standard filename of `trg_dsm2_qtd_stp2_config.version.txt` that will be read when configuring.