Rules for creating Register and Dictionary entries in Tier1 .dat files

John Nelson

18 December 2008, updated 8 January 2009

Modifications: 8 January: Additional optional field for dictionary definitions.

The dictionary file used for Run Control contains four (or optionally, five) fields:

<crate object number> <VME board sub-address> <register number> <name> [#comment]

The <crate object number> is the crate identifier and is defined in RC_Config.h. The <VME board sub-address> is the MS byte of the board's actual VME address translated into decimal. The <register number> relates to the register number in a given DSM or QT board. The dictionary name is <name> and is a single string, that is, without spaces

An additional, optional, field has been added to the dictionary entry: a comment which must be prefaced with the # character. The default register value and the comment will appear on the RC page

For a given DSM board, the register number is unambiguous. Register numbers range from 0 upwards.

Register numbers on QT boards have multiple assignments since each QT board consists of 5 parts (a mother board and 4 daughter boards) each of which has 64 registers with numbers in the range 0 to 63. In order to distinguish between these, <register number> in the dictionary file will have the decimal form: Axx where xx is the decimal register number (0 to 63) and A = 0,1,2,3,4 or 5 depending on whether the register number refers to a mother board (0); a named daughter board (1,2,3 or 4) or all 4 daughter boards (5).

NB: An important change has been introduced into Tier1 file definitions. Registers are referred to by register *number* and **not** by register *address*.

DSM crate definitions

1) If a board definition is preceded by a short name which begins with two ## characters such as:

```
##BE003
DSM_BASE_ADDRESS 0x12000000
etc.
```

then the short name will be inserted into the dictionary file to make it more readable.

2) Registers are defined by implied register *number* from 0 upwards. As an example:

```
DSM_ENG_REG 5
0x0b   0 BEMC-HighTowerTh0
0x0e   1 BEMC-HighTowerTh1
0x12   2 BEMC-HighTowerTh2   #This is threshold 2 for the High Tower
0x11   3 BEMC-HighTowerTh3
0x19   4 BEMC-TriggerPatchTh0
```

defines 5 registers in the range 0 to 4. Register values are in the first column and may be in hexadecimal or decimal format. (*Note:* the second column in DSM Tier files previously defined a bit-mask. This is no longer required and should not be included in the definitions.)

Dictionary entries follow on each line specifying the explicit register number *which must be in decimal format*, and the name to be assigned to that register in the dictionary file. Optionally, a comment string can be added after the label. This string *must* be prefixed with the # character.

If the explicit register number in the dictionary entry is -1 as in the next example:

```
0x19  -1 BEMC-TriggerPatchTh0
```

then the name will *not* be entered in the dictionary but serves as a reference to the reader of the crate definition file. Any default value or comment string will be ignored.

QT crate definitions

1) If a board definition is preceded by a short name which begins with two ## characters such as:

```
##QT003
QT_BASE_ADDRESS 0x12000000
etc.
```

then the short name will be inserted into the dictionary file to make it more readable.

2) Each QT board has a mother board and 4 daughter boards each with up to 64 registers that may be defined. Registers may be defined in any order by giving the register number in the range 0 to 63, and its value.

Register numbers and values may be given in hexadecimal or decimal format but not in mixed format on the same line.

As an example:

```
QT_MB_REG 3
0x1 0x36      1   Gate_Start_Delay
15  61       15   GateEndDelay       #The Gate End value should not exceed 10000
0x2 0x32      2   Output_Latch_Delay
```

defines 3 mother board registers with numbers 1,15 and 2, two of which are defined in hexadecimal. The corresponding values are given. The following line definition will lead to a scanning error:

```
1  0x36      1    Gate_Start_Delay
```

since the register number and value are in mixed decimal and hexadecimal format.

Dictionary entries follow on each line specifying the explicit register number *which must be in decimal format*, and the name to be assigned to that register in the dictionary file. Optionally, a comment string can be added after the register label. This string *must* be prefixed with the # character.

If the explicit register number in the dictionary entry is -1 as in this example:

```
0x2 0x32    -1 OutputLatchDelay
```

then the name will *not* be entered in the dictionary but serves as a reference to the reader of the crate definition file.

In this example, all 4 daughter boards on a particular QT board will be set as follows:

```
QT_DB_REG 2
3   1          3   Do_not_use_LUT
0x2 0x9        2   Start_writing_at-offset_9
```

two registers are defined, namely 3 and 2, and the corresponding values will be stored in all 4 daughter boards.

Another example:

```
QT_D3_REG 2
3   1          3   Do_not_use_LUT
2   9          2   Start_writing_at_offset_9
```

defines two registers for daughter board 3 only.

Multiple (wild-card) definitions

In certain circumstances, such as with QT crates, it is useful to be able to define a set of registers values that will loaded into a particular set of registers for all boards in a given crate. These definitions cannot be included in the Tier1 .dat files.

A separate wild-card file must be created and stored in the Tier1/Dictionary directory. The name of this file *must* be the same as the master file defining a given Tier1 binary file. The contents of the wild-card file will be appended to the dictionary file when it is created during a MakeConfig session.

QT definitions

The rules for constructing these are as follows. There are four fields:

<special object> <qt object number> <register number> <name>

The <special object> must *always* be 29. If the <qt object number> is a xxx_QT_OBJECT as defined in RC_Config.h then what follows refers only to that particular crate. The <register number> will be in the QT format, namely Axx. For example:

```
# QT Boards
#
29 11 1  QT-1-GateStart        31
29 11 2  QT-1-OutputOffset   29  This is the output offset
29 11 15 QT-1-GateStop
```

will ensure that at Run Start, mother board registers (Axx format, A=0) 1,2 and 15 will be loaded for all mother boards in the crate defined by <qt object number> 11. These entries refer to the decimal *number* of a register as defined in December release of QT_mem_map.pdf.

In the case of wild card definitions, the default register and comment string can be added as shown. Note thar in the case of wild-card definitions, the comment string must *not* have the # character as used in normal definitions and the comment string *must* be preceded by the default register value. Adding the default register value without the comment string is allowed.

Similarly:

```
29 12 103  Do_not_use_LUT
```

will set register 3 of all daughter boards number 1 in <qt object number> 12.

There are two other assignments to <qt object number> and these are 128 and 129.

The following construction:

```
29 128 5   QT-RunMode
29 128 13 QT-ZeroSuppress
```

will load registers 5 and 13 into all mother boards in all QT crates. Similarly:

```
29 129 2 QT-DataOffset
29 129 3 QT-UseLUT
```

will load registers 2 and 3 into all daughter boards in all QT crates.

TCU Input Bit Definitions

The file containing the <wild-card numbers> must also contain the definition of the TCU input bits. The format to be used is:

<special object> <0> <bit number> <description> where <special object> *must* be 32 and the second field *must* be 0. This should have a short header as shown in the following example:

```
#
# TCU Bit definition
```

#
32 0 0 MTD
32 0 1 TOF
32 0 2 FMS-pre
32 0 3 FMS
32 0 4 FPDE
32 0 5 ETOT
32 0 6 BHT-bit0
32 0 7 BHT-bit1
32 0 8 EJP0
32 0 9 JP1
32 0 10 JP2
32 0 11 FMSLED
32 0 12 VPD
32 0 13 BBC
32 0 14 BBC-pre
32 0 15 ZeroBias