Register Definitions for DSMs and QTs

John Nelson

5 December 2007

In October 2002, changes were made to the definitions of registers loaded into the Tier1 files. The definitions are to be found in RC_Config.h. On 4 December 2007, some changes were introduced into this scheme and this document provides an uptodate description.

An array `RegValue registers[1500]` (terminated by a zero entry) is loaded into the Tier1 file where the definition of `RegValue` is given by:

```
struct RegValue
{
  UINT32 object;
  UINT32 index;
  UINT32 reg;
  UINT32 value;
};
```

where each DSM/QT crate has an identifier as follows:

```
#define L1_DSM_OBJECT    1
#define BC1_DSM_OBJECT   2
#define MIX_DSM_OBJECT   3
#define CTB_DSM_OBJECT   4
#define BCW_DSM_OBJECT   5
#define BCE_DSM_OBJECT   6
#define EEC_DSM_OBJECT   7
#define BBC_DSM_OBJECT   8
#define FPE_DSM_OBJECT   9
#define FMS_DSM_OBJECT  10
#define QT1_FMS_OBJECT  11
#define QT2_FMS_OBJECT  12
#define QT3_FMS_OBJECT  13
#define QT4_FMS_OBJECT  14
```

Each DSM/QT board is identified by its 8-bit `index` which the most significant byte of the VME address of the board.

The item `reg` is, in the case of DSMs, the register number counting from 0,1,2 etc while, for historical reasons in the case of QTs, `reg` contains the least significant byte of the *address* of the register. (When QT configuration files were first defined, the least significant byte of the register *address* was inserted in the definition file and not the register *number*.)

The actual value to be loaded into the relevant register is specified by `value.`

It is important to note that the table of registers shown in the Run Control GUI is listed first by `object` number and within that context, by `index` number, shown in decimal. As usual, any register in the GUI list with value -1 will be ignored when registers are loaded.

<u>QT Boards</u>

The QT boards present particular problems, since each QT board consists of 5 sub-boards: a mother-board and 4 daughter-boards.

In order to accommodate QTs within the definition of `RegValue`, the definition of `index` has been extended in two quite different ways.

Individual QT board

The `index` should be treated as a hexadecimal entity as 0x00QV where:

V: is the most significant byte of the VME address of the board (as before) and
Q: takes values 0,1,2,3,4 or 5 where
Q=0: the register to be loaded is on the mother-board
Q=1,2,3 or 4: the register to be loaded is on daughter-board 1,2,3 or 4
Q=5: the register *value* will be loaded into the registers specified by `reg` on <u>all</u> daughter-boards on the particular QT board.

Multiple QT boards

The total number of registers available in the QT crates is 15360. Provision has been made to simplify the loading of multiple boards with the same register value.

If multiple loading is to take place, the item `object` will *not* refer to a particular `QTx_FMS_OBJECT` but will take a special value defined by `TRG_OBJECT = 29.`

The loading of registers is driven, in this case, by a different definition of `index` as follows:

`index = 128:` The mother-board register at address `reg` on <u>all</u> mother-boards in <u>all</u> crates will be loaded with `value.`
`index = 129:` The daughter-board register at address `reg` on <u>all</u> daughter-boards in <u>all</u> crates will be loaded with `value.`
`index = 11,12,13 or 14:` The mother-board register at address `reg` on <u>all</u> mother-boards in the specific crate with `QTx_FMS_OBJECT` number given by `index` will be loaded with `value.`
`index = 21,22,23 or 24:` The daughter-board registers at address `reg` on <u>all</u> daughter-boards in the specific crate with `QTx_FMS_OBJECT` number given by `(index - 10)` will be loaded with `value.`

Loading order of registers

1. All the registers defined in the QT configuration files which are part of the Tier1 file are loaded.
2. All QT registers defined by multiple loading in the Run Control GUI, that is, those with `TRG_OBJECT = 29 are` loaded next.
3. Finally, all individual QT registers defined in the Run Control GUI (those which are not `TRG_OBJECT = 29` items) will loaded, except, of course, those registers with value set to -1.