```
TITLE "VME Local Registers";

INCLUDE "vmemap.inc";

%**********************************************************************

This block contains all of the register used for the THC. The registers are:

Status          Addr 0 - The status register contains the current information
                            in the other registers. It is read only and has
                            16-bits worth of information. The bit definition is

                            Status[3..0] =  Emu status;
                                status0 =   emu_enabled;
                                status1 =   emu continuous;
                            Status[7..4] =  Clk status;
                                status4 =   SSTEP;
                                status5 =   LOC_CLK;
                                status6 =   CLK_ERR;
                                status7 =   CLK_GONE (this value not writable via
                                                clock register);
                            Status[10..8] = IRQ level to be used for interrupts;
                            Status[15..11] = Ctrl status;
                                Status11 =  GB Sync error;
                                Status12 =  Init_ack;

The following are write only registers

Control         Addr 0 - The control register is a write only register containing
                            board control information. 5-bit register.
Clk set         Addr 1 - This register sets bits regarding clock settings. 4-bit
                            register
Clk reset       Addr 2 - And this resets the bits for the clock settings.
Emu set         Addr 3 - Emulation setting bits are set here. 4-bit register
Emu reset       Addr 4 - And reset here
Emu pulse       Addr 5 - This will output a trigger pulse that will be available
                            at the front panel. No data is required. Writing to
                            the address will cause the pulse.
Irq             Addr 6 - Contains a 3-bit code that is equivalent to the IRQ level
                            that interrupts will be issued at.

Remember that since there is no vme address 0, the actual vme addresses used will
    be a multiple of 2, i.e the emu reset register would be addressed as 8 from
    vme.

**********************************************************************%

SUBDESIGN VmeRegs
```

```
(
    VmeD[15..0]         : BIDIR;    % VME Data bus.                    %

    RFCLK               : INPUT;    % 53 MHz Clock                     %
    WrInp               : INPUT;    % VME Write                        %
    RdInp               : INPUT;    % VME Read                         %
    VmeA[7..1]          : INPUT;    % VME Address bus.                 %
    VmeRegsel           : INPUT;    % VME Interface Register select    %
    /reset              : INPUT;    % reset signal %
    CLK_GONE            : INPUT ;   % If the rfclk dies, this bit goes active high.
                                        Put it's status in the clock register %
    CLK_ERR             : INPUT ;   % if rfclk misses a pulse (or 2) this signal goes active %
    SyncErr             : INPUT;    % The GB sync error from the serial link %
    Init_Ack            : INPUT;    % OR of the init acknowledges from the 128 sections %
    ExtTrig             : INPUT;   % External trigger that will start up emulation mode %
    ExtSnglClk          : INPUT;    % External single step clock %
    SSTEP               : OUTPUT ;  % Single step mode. Active high. This, in
                                        conjunction with the external sstep_clk allows
                                        an off board clock to run the board %
    LOC_CLK             : OUTPUT ;  % If this bit's set, and sstep mode isn't enabled,
                                         the onboard 53 mhz oscillator will take over
                                         and start issuing rfclk and bsync clocks %
    SSTEP_CLK           : OUTPUT;   % single step clock generated either by the ExtSnglClk or a write
                                        to the clock register's sstep clock bit %
    IRQSel[2..0]        : OUTPUT;   % The IRQ level that the THC will issue -
                                        writeable and readable from the IRQ
                                        register %
    VmeDtack            : OUTPUT;   % VME Data Acknowledge             %
    % Emulation Sequence Controller                              %
    EmuEnabled          : OUTPUT;   % Emulation Run Enabled     %
    EmuTrig             : OUTPUT;   % Start Emulation pulse     %
    EmuContin           : OUTPUT;   % Run continuous emulation  %

)
VARIABLE
    StatReg[15..0]      : TRI;      % VME Status Register       %
    EmuReg[3..0]        : DFF;      % Emulation Control         %
    ClkReg[3..0]        : DFF;      % Clock Control             %
    CtrlReg[4..0]       : DFF;      % Control Register          %
    IRQReg[2..0]        : DFF;      % Interrupt request level register %
    WrStb               : node;     % Write Strobe              %
    RdEnable            : node;     % Read Enable               %
    DelayLn[7..0]       : DFF;      % Delay Line for data strobe and dtack %
    ExtFF               : DFF;      % Flop to latch the external trigger input %
    ClkErrLatch         :DFF;       % flop to latch the clk_err bit %

        % used a gray code counter to eliminate glitches %
    ClearReg    : MACHINE
```

```
        OF BITS (st1,st2,st3)
        WITH STATES
            (NoOp                   = B"000",
            ChkRdEnable         = B"001",
            Wait4Dtack              = B"011",
            Wait4DtackInactive      = B"010",
            Wait4NoVmeRegsel        = B"110",
            n5                      = B"111",
            n6                      = B"101",
            n7                      = B"100"
            );


BEGIN

    % power up / reset conditions %
    EmuReg[].clrn = /reset;
    ClkReg[1..0].clrn = /reset;
    ClkReg[2].clrn = /reset and !VmeDtack;
    ClkReg[3].clrn = /reset;
    IRQReg[].clrn = /reset;
    ExtFF.clrn = /reset AND !EmuTrig;
    ClkErrLatch.clrn = /reset;

    ClearReg.clk = RFCLK;

        CASE ClearReg IS
            WHEN NoOp =>    % wait for the ram select bit to be active before we do anything %
                IF (ClkErrLatch.q == GND) THEN
                    ClearReg = NoOp;
                ELSE
                    ClearReg = ChkRdEnable;
                END IF;
            WHEN ChkRdEnable => % wait here until either the read or write line goes active %
                IF (RdEnable == VCC) THEN
                    ClearReg = Wait4Dtack;
                ELSE
                    ClearReg = ChkRdEnable;
                END IF;
            WHEN Wait4Dtack =>
                IF (VmeDtack == GND) THEN
                    ClearReg = Wait4Dtack;
                ELSE
                    ClearReg = Wait4DtackInactive;
                END IF;
            WHEN Wait4DtackInactive =>
                IF (VmeDtack == VCC) THEN
                    ClearReg = Wait4DtackInactive;
```

```
            ELSE
                ClearReg = Wait4NoVmeRegsel;
            END IF;
        WHEN Wait4NoVmeRegsel =>
            IF (VmeRegsel == VCC) THEN
                ClearReg = Wait4NoVmeRegsel;
            ELSE
                ClkErrLatch.clrn = GND;
                ClearReg = n5;
            END IF;
        WHEN n5 =>
            ClearReg = n6;
        WHEN n6 =>
            ClearReg = n7;
        WHEN n7 =>
            ClearReg = NoOp;
    END CASE;


% External trigger flop %
ExtFF.clk = ExtTrig;
ExtFF.d = VCC;
%Clock error latch %
ClkErrLatch.d = VCC;
ClkErrLatch.clk = CLK_ERR;
% Write Strobe %
WrStb = WrInp & VmeRegsel & DelayLn[4] & !DelayLn[7];

% Delay Line SR %
DelayLn[].clk   = RFCLK;
DelayLn[].clrn = VmeRegsel AND /reset;
DelayLn[0].d = VCC;
DelayLn[7..1].d = DelayLn[6..0].q;

% The Emulation register.       %
EmuReg[].clk = (WrStb AND VmeRegsel AND ((VmeA[7..1] == EmuSet) OR (VmeA[7..1] == EmuRes)));

% The Clock register.       %
ClkReg[].clk = (WrStb AND VmeRegsel AND ((VmeA[7..1] == ClkSet) OR (VmeA[7..1] == ClkRes)));
% The Control register      %
CtrlReg[].clk = (WrStb AND VmeRegsel AND (VmeA[7..1] == CtrlAd));
% The IRQ register      %
IRQReg[].clk = (WrStb AND VmeRegsel AND (VmeA[7..1] == IRQAd));

% The contents of the status register %
StatReg[].oe =          RdEnable;
StatReg[3..0].in =      EmuReg[3..0].q;
StatReg[5..4].in =      ClkReg[1..0].q;
```

```
    StatReg[6].in =         ClkErrLatch.q;
    StatReg[7].in =         CLK_GONE;
    StatReg[10..8].in =     IRQReg[2..0].q;
    StatReg[11].in =        SyncErr;
    StatReg[12].in =        Init_Ack;
    StatReg[15..13].in =    CtrlReg[2..0].q;

    CASE VmeA[7..1] IS
        WHEN EmuSet =>
            EmuReg[].d = (EmuReg[].q OR VmeD[3..0]);
        WHEN EmuRes =>
            EmuReg[].d   = (EmuReg[].q AND !VmeD[3..0]);
        WHEN ClkSet =>
            ClkReg[].d = (ClkReg[].q OR VmeD[3..0]);
        WHEN ClkRes =>
            ClkReg[].d   = (ClkReg[].q AND !VmeD[3..0]);
        WHEN CtrlAd =>
            CtrlReg[].d = VmeD[4..0];
        WHEN IRQAd =>
            IRQReg[].d = VmeD[2..0];
    END CASE;
    RdEnable = VmeRegsel AND RdInp AND (VmeA[7..1] == StatAd) ;

    % Status Output %
    VmeD[15..0] = StatReg[15..0].out;

    %Emulation register definition %
    EmuEnabled  = EmuReg[0].q;
    EmuContin   = EmuReg[1].q;
    EmuTrig = ((ExtFF.q) OR (WrStb AND VmeRegsel AND (VmeA[7..1] == EmuPls))) AND EmuEnabled;
    % IRQ register definition %
    IRQSel[] = IRQReg[].q;
    % clock register definition %
    SSTEP = ClkReg[0].q AND EmuEnabled AND !LOC_CLK;
    LOC_CLK = ClkReg[1].q AND EmuEnabled AND !SSTEP;
    % Generate the single step clock %
    SSTEP_CLK = EmuEnabled AND SSTEP AND (ExtSnglClk OR ClkReg[2].q);

    VmeDtack = DelayLn[6] OR (VmeDtack AND VmeRegsel);

END;
```