

```

// *****
// **
// ** DAC7715.v - DAC7715 12-BIT SERIAL DAC (VCC = +15.0V)
// **
// *****
// **
// **          COPYRIGHT (c) 2003 YOUNG ENGINEERING
// **          ALL RIGHTS RESERVED
// **
// ** THIS PROGRAM IS CONFIDENTIAL AND A TRADE SECRET OF YOUNG ENGINEERING. THE RECEIPT OR
// ** POSSESSION OF THIS PROGRAM DOES NOT CONVEY ANY RIGHTS TO REPRODUCE OR DISCLOSE ITS
// ** CONTENTS, OR TO MANUFACTURE, USE, OR SELL ANYTHING THAT IT MAY DESCRIBE, IN WHOLE OR IN
// ** PART, WITHOUT THE SPECIFIC WRITTEN CONSENT OF YOUNG ENGINEERING.
// **
// *****
// ** Revision      : 1.0
// ** Modified Date  : 08/01/2003
// ** Revision History:
// **
// ** 08/01/2003: Initial design
// **
// *****
// **          TABLE OF CONTENTS
// **          *****
// **-----**
// **  DECLARATIONS
// **-----**
// **-----**
// **  INITIALIZATION
// **-----**
// **-----**
// **  CORE LOGIC
// **-----**
// **-----**
// **  TIMING CHECKS
// **-----**
// **
// *****

```

```
`timescale 1ns/10ps
```

```
module DAC7715 (VOUTA, VOUTB, VOUTC, VOUTD, SDI, CLK, CS_N, LOADDAC_N, LOADREG_N, RESET_N, RESETSEL, RESET);
```

```

input          SDI;          // serial data in
input          CLK;          // serial data clock

input          CS_N;         // chip select

input          LOADDAC_N;    // DAC load strobe - all
input          LOADREG_N;    // DAC register load strobe

input          RESET_N;     // DAC reset strobe
input          RESETSEL;    // reset command select

input          RESET;       // system reset

output [11:00] VOUTA;       // DAC A output voltage
output [11:00] VOUTB;       // DAC B output voltage
output [11:00] VOUTC;       // DAC C output voltage
output [11:00] VOUTD;       // DAC D output voltage

```

```

// *****
// **  DECLARATIONS
// ** *****

```

```

reg [15:00] ShiftRegister; // serial shift register
wire ShiftClock;          // shift register clock

reg [11:00] InputRegisterA; // DAC A input register
reg [11:00] InputRegisterB; // DAC B input register
reg [11:00] InputRegisterC; // DAC C input register
reg [11:00] InputRegisterD; // DAC D input register

reg [11:00] VOUTA;         // DAC A output voltage
reg [11:00] VOUTB;         // DAC B output voltage
reg [11:00] VOUTC;         // DAC C output voltage
reg [11:00] VOUTD;         // DAC D output voltage

```

```
// *****
```

```

// **  INITIALIZATION  **
// *****

initial begin
    VOUTA = 12'hXXX;
    VOUTB = 12'hXXX;
    VOUTC = 12'hXXX;
    VOUTD = 12'hXXX;
end

// *****
// **  CORE LOGIC  **
// *****

assign ShiftClock = CLK | CS_N;

always @(posedge ShiftClock) begin
    ShiftRegister[00] <= SDI;
    ShiftRegister[01] <= ShiftRegister[00];
    ShiftRegister[02] <= ShiftRegister[01];
    ShiftRegister[03] <= ShiftRegister[02];
    ShiftRegister[04] <= ShiftRegister[03];
    ShiftRegister[05] <= ShiftRegister[04];
    ShiftRegister[06] <= ShiftRegister[05];
    ShiftRegister[07] <= ShiftRegister[06];
    ShiftRegister[08] <= ShiftRegister[07];
    ShiftRegister[09] <= ShiftRegister[08];
    ShiftRegister[10] <= ShiftRegister[09];
    ShiftRegister[11] <= ShiftRegister[10];
    ShiftRegister[12] <= ShiftRegister[11];
    ShiftRegister[13] <= ShiftRegister[12];
    ShiftRegister[14] <= ShiftRegister[13];
    ShiftRegister[15] <= ShiftRegister[14];
end

always @(LOADREG_N or RESET_N or RESETSEL or ShiftRegister) begin
    if (RESET_N == 0) begin
        if (RESETSEL == 0) begin
            InputRegisterA = 12'h000;
            InputRegisterB = 12'h000;
            InputRegisterC = 12'h000;
            InputRegisterD = 12'h000;
        end
        else begin
            InputRegisterA = 12'h800;
            InputRegisterB = 12'h800;
            InputRegisterC = 12'h800;
            InputRegisterD = 12'h800;
        end
    end
    else if (LOADREG_N == 0) begin
        case (ShiftRegister[15:14])
            2'b00 : InputRegisterA = ShiftRegister[11:00];
            2'b01 : InputRegisterB = ShiftRegister[11:00];
            2'b10 : InputRegisterC = ShiftRegister[11:00];
            2'b11 : InputRegisterD = ShiftRegister[11:00];
        endcase
    end
end

always @(LOADDAC_N or RESET_N or RESETSEL or
    InputRegisterA or InputRegisterB or
    InputRegisterC or InputRegisterD) begin
    if (RESET_N == 0) begin
        if (RESETSEL == 0) begin
            VOUTA = 12'h000;
            VOUTB = 12'h000;
            VOUTC = 12'h000;
            VOUTD = 12'h000;
        end
        else begin
            VOUTA = 12'h800;
            VOUTB = 12'h800;
            VOUTC = 12'h800;
            VOUTD = 12'h800;
        end
    end
    else if (LOADDAC_N == 0) begin
        VOUTA = InputRegisterA[11:00];
        VOUTB = InputRegisterB[11:00];
        VOUTC = InputRegisterC[11:00];
    end
end

```

```

        VOUTD = InputRegisterD[11:00];
    end
end

// *****
// **   TIMING CHECKS   **
// *****

specify
specparam
    tCH = 30,           // CLK pulse width - high
    tCL = 50,           // CLK pulse width - low
    tDS = 25,           // SDI to CLK setup time
    tDH = 20,           // SDI to CLK hold time
    tCSS = 55,          // CS_N to CLK setup time
    tCSH = 15,          // CS_N to CLK hold time
    tLD1 = 40,          // LOADREG_N to CLK setup time
    tLD2 = 15,          // LOADREG_N to CLK hold time
    tLDDW = 45,         // LOADDAC_N pulse width - low
    tLDRW = 45,         // LOADREG_N pulse width - low
    tRSSH = 25,         // RESETSEL to RESET_N setup
    tRSTW = 70;         // RESET_N pulse width - low

    $width (posedge CLK, tCH);
    $width (negedge CLK, tCL);
    $width (negedge LOADDAC_N, tLDDW);
    $width (negedge LOADREG_N, tLDRW);
    $width (negedge RESET_N, tRSTW);

    $setup (SDI, posedge CLK &&& (RESET==0), tDS);
    $setup (CS_N, posedge CLK &&& (RESET==0), tCSS);
    $setup (LOADREG_N, posedge CLK &&& (RESET==0), tLD1);

    $setup (RESETSEL, negedge RESET_N &&& (RESET==0), tRSSH);

    $hold (posedge CLK &&& (RESET==0), SDI, tDH);
    $hold (posedge CLK &&& (RESET==0), CS_N, tCSH);
    $hold (posedge CLK &&& (RESET==0), LOADREG_N, tLD2);
endspecify

endmodule

```