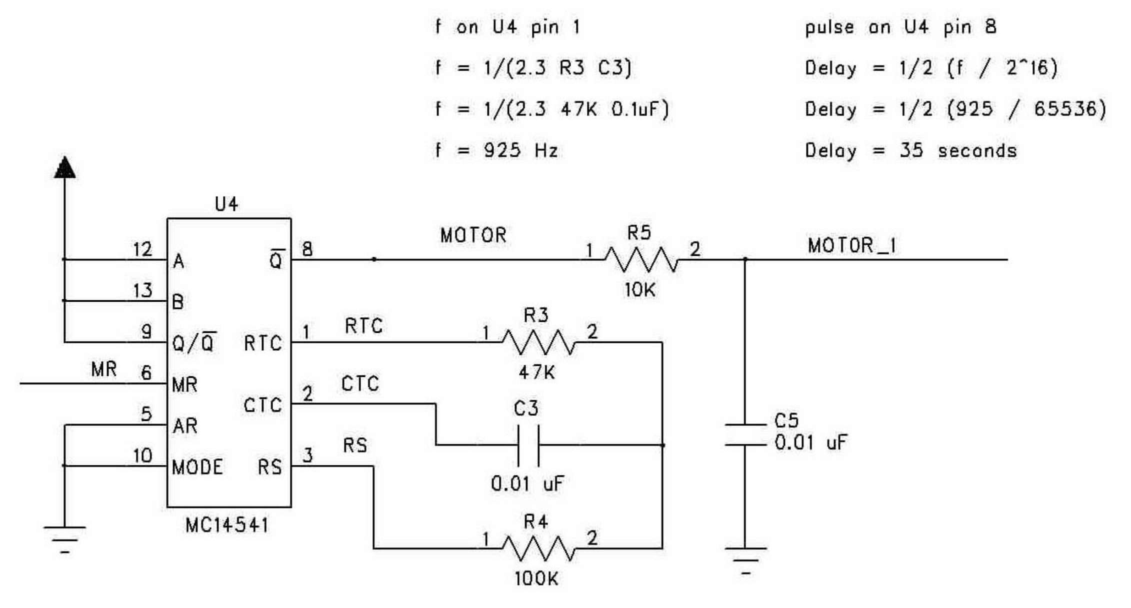
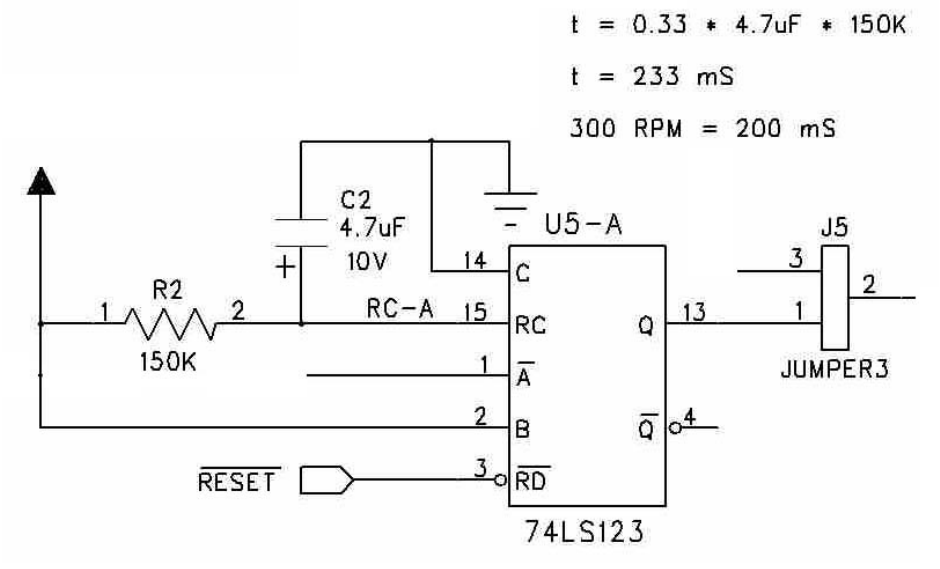
DC5 Design Description

The two major components in the DC5 are U1, the WD2797 Floppy Disk Controller, and U2, the Xilinx XC9572 programmable logic device. The operation of the WD2797 is provided in a separate data sheet. The operation of the XC9572 is given below.

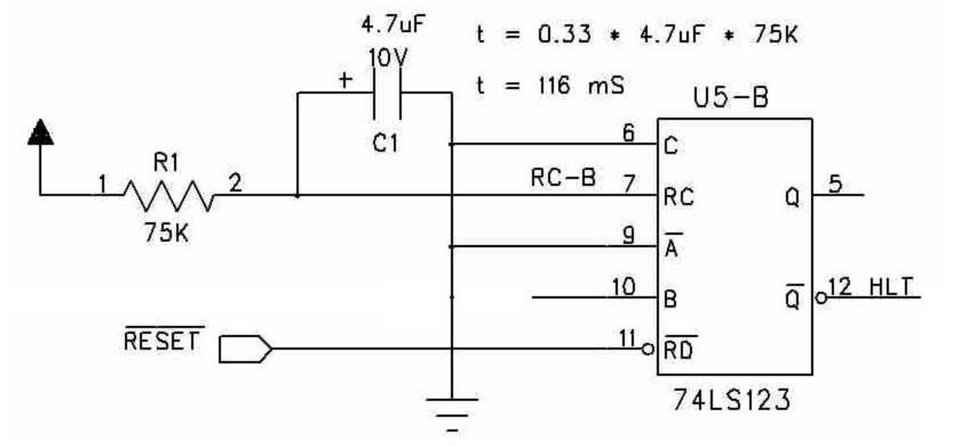
The motor run timer is triggered (U4 pin 6) whenever the WD2797 FDC is accessed (8018-801B or E018-E01B). The motor will run for about 30 seconds after the last access. The time is controlled by the values of R3, R4 and C3. See the MC14541 data sheet for details on adjusting the time. The output of this timer is also used for the Ready signal in the "00" setting of J5. The motor control is from U4 pin 8 while R5 and C5 provide a delay for the Ready signal.



The index pulse detector is triggered (U5 pin 1) by the IP signal from the floppy drive. At 300 RPM this pulse occurs every 200 ms. The detector is set at 233 ms and its output will be high as long as the disk is spinning. (Newer 3.5 inch drives mask the IP signal when the drive is changing tracks and the output of U5A will go low.) The output of U5A is use in the "09" setting of J5.



The head load timer is triggered (U5 pin 10) when the HLD signal from U1 goes high. The output on U5 pin 12 goes low for about 100 ms then stays high. This is connected to the HLT pin on U1. The head load feature is for 8-inch disk drives and this circuit satisfies the delay required by the WD2797 FDC.



The pull-up resistors R8, R9, R10 and R11 are 150 ohms as specified in for the original 5.25 floppy drives. These can be replaced with 330 ohms or 1000 ohms resistors if desired. Current production drives use 1000 ohms pull-up resistors but work with 150 ohms.

The output buffers U7 and U8 are 74LS06 open-collector inverters that could be replaced with 74LS04 or 74HC04 inverters. The floppy drives are designed for open-collector parts but many new designs use a standard inverter.

C15 is an optional 100 pf capacitor that was used on many SWTPC designs. It is not required.

The voltage regulators in the lower right hand corner are only used with the FD1771 chip.

The capacitor C6 is mislabeled on the silk screen as C5. C6 is above U1, WD2797, and to the left of R6. The real C5 is next to R5.

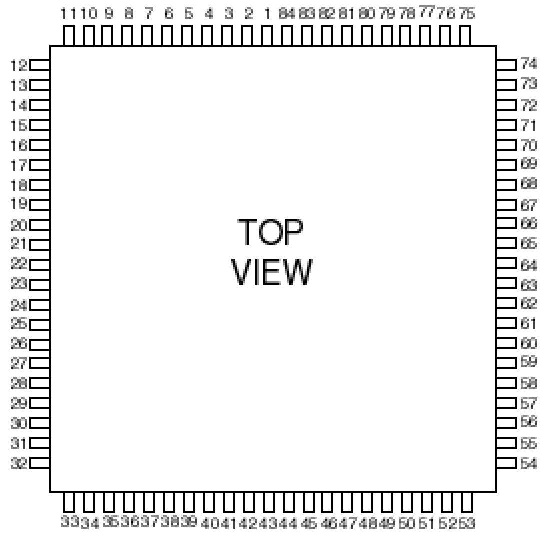
The optional jumper to the left of the U9 is to disable the output of the oscillator. This is not required.

The XC9572 design is described in the Verilog hardware description language. It is similar to the “C” programming language and the ABEL PLD language. A complete language description is provided with the Xilinx design software.

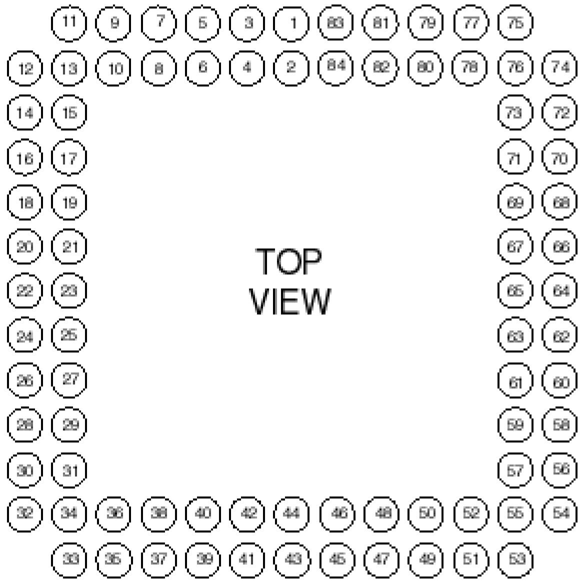
XC9572 Design Description

```
                                                                            F
                                                                            i
              F           F  F  F  F  F  F  F                       M     v
              D           D  D  D  D  D  D  D                       R  P  e
              C           C  C  C  C  C  C  C                       _  u  _
              _     M     _  _  _  _  _  _  _     I                 T  l  E
              D     o     D  D  D  D  D  D  D  P  N  P     P  D     i  s  i
              <     t  G  <  <  <  <  <  <  <  G  T  G  D  G  D  V  m  e  g
              0  A  o  N  1  2  3  4  5  6  7  N  R  N  R  N  E  C  e  I  h
              >  1  r  D  >  >  >  >  >  >  >  D  Q  D  Q  D  N  C  r  n  t
              --------------------------------------------------------------
             /11 10 9  8  7  6  5  4  3  2  1  84 83 82 81 80 79 78 77 76 75 \
        Osc | 12                                                          74 | Reset
         RE | 13                                                          73 | VCC
     CS_FDC | 14                                                          72 | FDC_Clk
         WE | 15                                                          71 | PGND
        GND | 16                                                          70 | SSO
         A0 | 17                                                          69 | PGND
   Mode_416 | 18                                                          68 | Drive1
       PGND | 19                                                          67 | PGND
Mode_35Disk | 20                                                          66 | Drive0
  Mode_2Reg | 21                       XC9572-15-PC84                     65 | PGND
        VCC | 22                                                          64 | VCC
       PGND | 23                                                          63 | PGND
   Mode_Cal | 24                                                          62 | Drive3
       PGND | 25                                                          61 | PGND
       PGND | 26                                                          60 | GND
        GND | 27                                                          59 | TDO
        TDI | 28                                                          58 | Drive2
        TMS | 29                                                          57 | PGND
        TCK | 30                                                          56 | Motor_On
       PGND | 31                                                          55 | IRQ
  IO_Select | 32                                                          54 | Side_Select
            \ 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 /
              --------------------------------------------------------------
              R  E  S  S  S  V  S  S  S  G  S  P  S  P  R  R  G  P  R  P  R
              W     S  S  S  C  S  S  S  N  S  G  S  G  S  S  N  G  S  G  S
                    3  3  3  C  3  3  3  D  3  N  3  N  0  1  D  N  3  N  2
                    0  0  0     0  0  0     0  D  0  D           D     D
                    _  _  _     _  _  _     _     _
                    D  D  D     D  D  D     D     D
                    <  <  <     <  <  <     <     <
                    7  6  5     4  3  2     1     0
                    >  >  >     >  >  >     >     >
```

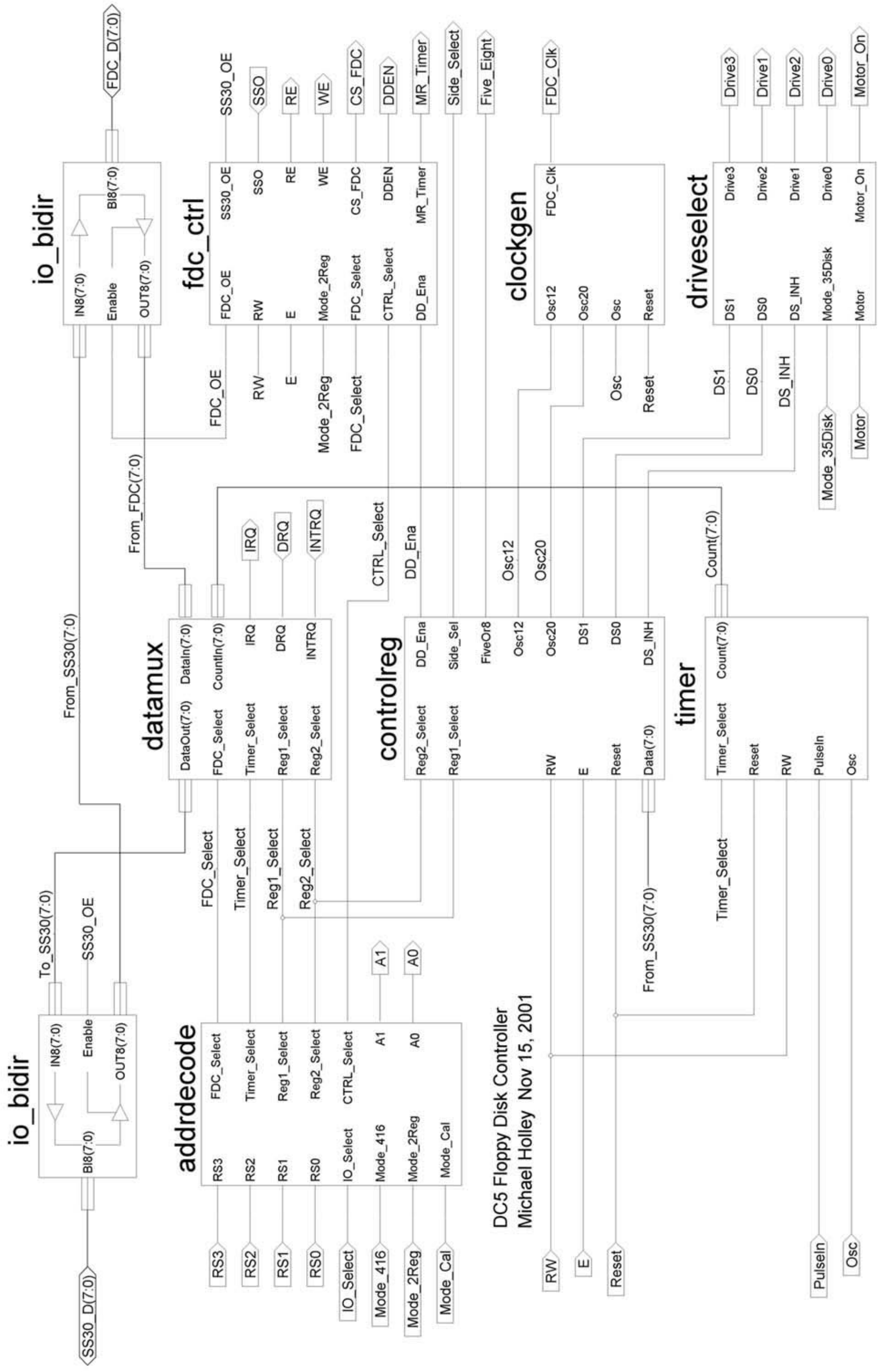
PLCC Package



PGA Socket Footprint
(13 x 13 x 2)



This footprint is from Emulation Technology, Inc.
http://www.emulation.com/footprints/84-PCC5.gif


io_bidir
IN8(7:0)
Enable
OUT8(7:0)
BI8(7:0)
FDC_D(7:0)
SS30_D(7:0)
fdc_ctrl
FDC_OE
RW
E
Mode_2Reg
FDC_Select
CTRL_Select
DD_Ena
SS30_OE
SSO
RE
WE
CS_FDC
DDEN
MR_Timer
Side_Select
Five_Eight
clockgen
Osc12
Osc20
Osc
Reset
FDC_Clk
driveselect
DS1
DS0
DS_INH
Mode_35Disk
Motor
Drive3
Drive2
Drive1
Drive0
Motor_On
datamux
DataOut(7:0)
DataIn(7:0)
CountIn(7:0)
Timer_Select
Reg1_Select
Reg2_Select
IRQ
DRQ
INTRQ
From_SS30(7:0)
From_FDC(7:0)
To_SS30(7:0)
controlreg
Side_Sel
FiveOr8
Data(7:0)
timer
Count(7:0)
PulseIn
addrdecode
RS3
RS2
RS1
RS0
IO_Select
Mode_416
Mode_Cal
A1
A0
DC5 Floppy Disk Controller
Michael Holley Nov 15, 2001

Address Decoding

The original SWPTC 6800 systems had I/O slots with 4 address locations. The DC1 floppy disk controller plugged into slot 6 (8018 to 801B) but it needed 5 address locations. To do this you had to jumper the adjacent I/O select (slot 5 – 8014 to 8017) to the UD3 line. The SWTPC 6809 systems mapped slot 6 to E018 to E01B and mapped slot 5 to E014 to E017.

The SWTPC 6809 systems could be configured with 16 address locations for each I/O slot. The UD3 and UD2 lines became the RS2 and RS3 lines. Because 6809 systems could have 4 or 16 address I/O slots the DC3 and DC4 floppy disk controllers had a 4 or 16 address jumper. In the 16-address location mode the card in I/O slot 1.

This card emulates this with the Mode_416 control pin. When this pin is high (SW1 position 3 OFF) the address decoding uses RS3 and RS2 as address lines. When the pin is low, RS2 is treated as an I/O select.

The 8014 to 8017 (E014 to E017) address range is used to write to the drive select control register and read the INTRQ and DRQ status. The 8018 to 801B (E018 to E01B) address range is used to read and write the WD2797 FDC chip (or calibration counter/timer).

```
module AddrDecode(RS3,RS2,RS1,RS0,IO_Select,Mode_416,Mode_2Reg,Mode_Cal,
                  FDC_Select, CTRL_Select, Reg1_Select, Reg2_Select, Timer_Select, A1,A0);

    input  RS3, RS2, RS1, RS0;
    input  IO_Select;
    input  Mode_416, Mode_2Reg, Mode_Cal;
    output FDC_Select, CTRL_Select;
    output Reg1_Select, Reg2_Select, Timer_Select;
    output A1, A0;

    wire E018_B, E014_7;
    wire E014, E015, E016, E017;

    assign E018_B =  Mode_416 & ~IO_Select &  RS3 & ~RS2
                  | ~Mode_416 & ~IO_Select;

    assign E014_7 =  Mode_416 & ~IO_Select & ~RS3 &  RS2
                  | ~Mode_416 & ~RS2;

    assign E014   =  Mode_416 & ~IO_Select & ~RS3 &  RS2 & ~RS1 & ~RS0
                  | ~Mode_416 & ~RS2                     & ~RS1 & ~RS0;

    assign E015   =  Mode_416 & ~IO_Select & ~RS3 &  RS2 & ~RS1 &  RS0
                  | ~Mode_416 & ~RS2                     & ~RS1 &  RS0;

    assign E016   =  Mode_416 & ~IO_Select & ~RS3 &  RS2 &  RS1 & ~RS0
                  | ~Mode_416 & ~RS2                     &  RS1 & ~RS0;

    assign E017   =  Mode_416 & ~IO_Select & ~RS3 &  RS2 &  RS1 &  RS0
                  | ~Mode_416 & ~RS2                     &  RS1 &  RS0;

    // Outputs
    assign CTRL_Select    = E014_7 | Timer_Select;

    assign FDC_Select     = E018_B & (Mode_Cal == 1'b1);

    assign Timer_Select   = E018_B & (Mode_Cal == 1'b0);

    assign Reg1_Select    = (Mode_2Reg == 1'b1) & E014_7 |
                            (Mode_2Reg == 1'b0) & (E014 | E015);

    assign Reg2_Select    = (Mode_2Reg == 1'b0) & (E016| E017);

    assign A1 = RS1;
    assign A0 = RS0;

endmodule
```

Control Registers

The SWTPC floppy disk controllers had one control register that selected the disk drive. It was located at I/O address 8014 on the 6800 systems and E014 on the 6809 systems. This register was not decoded to a single address; it used 4 address from E014 to E017 (8014 to 8017). Most FLEX software used just the single address so an additional control register can be added at E016-E017 (8016-8017).

The Mode_2Reg pin enables this second register. When the pin is low (SW1 position 2 ON) REG1 is at E014-E015 (8014-8015) and REG2 is at E016-E017 (8016-8017). When the pin is low REG1 is at E014 to E017 (8014 to 8017).

Control Register 1							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit0
DS_INH	Side_Sel	x	x	x	x	DS1	DS0

Control Register 2							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit0
x	x	DD_Ena	FiveOr8	Osc20	Osc12	x	x

The bits Osc20 and Osc12 control the clock divider circuit. The FiveOr8 is connected to the WD2797 5/8 pin.

```
module ControlReg(Reg1_Select, Reg2_Select, RW, E, Data, Reset,DS_INH,
                  Side_Sel, DD_Ena, FiveOr8, Osc20, Osc12, DS1, DS0);

   input  Reg1_Select, Reg2_Select;
   input  RW, E;
   input  [7:0] Data;
   input  Reset;
   output DS_INH, Side_Sel, DS1, DS0;
   output DD_Ena ,FiveOr8, Osc20,Osc12;

   wire Clk1, Clk2;
   reg DS_INH, Side_Sel, DS1, DS0;
   reg DD_Ena ,FiveOr8, Osc20,Osc12;

   // Control Register 1
   assign Clk1 =  Reg1_Select & ~RW & ~E;

   always @(posedge Clk1 or negedge Reset) begin
      if(Reset == 'b0)
         {DS_INH,Side_Sel,DS1,DS0} = 4'b0000;
      else
         {DS_INH,Side_Sel,DS1,DS0} = {Data[7],Data[6],Data[1],Data[0]};
   end

   // Control Register 2
   assign Clk2 =  Reg2_Select & ~RW & ~E;
   always @(posedge Clk2 or negedge Reset) begin
      if(Reset == 'b0)
         {DD_Ena,FiveOr8,Osc20,Osc12} = 4'b0000;
      else
         {DD_Ena,FiveOr8,Osc20,Osc12} = {Data[5],Data[4],Data[3],Data[2]};
   end

endmodule
```

Data Multiplexer

The FDC data and the INTRQ / DRQ status lines are multiplexed onto the SS30 data bus. During the read cycle when RW is high and the address is between E014 and E01B (8014 and 801B) the buffers to SS30 data bus are enabled. The FDC data is present except when the address is between E014 and E017 (8014 and 8017).

The SS30 data bus to routed to the FDC data pins. The buffers to the FDC are tri-state except when RW is low and the address is between E018 and E01B (8018 to 801B.)

Normally the status of DRQ and INTRQ are read between locations E014 and E017 (8014 and 8017). When in the 2-register mode, the design version number will be read at E016 and E017 (8016 and 8017). Revision 1 was only at E016 (8016)

When in the calibration mode the Counter/Timer is read instead of the FDC chip.

```
module DataMux(DataIn,CountIn,DRQ,INTRQ,
               FDC_Select, Reg1_Select, Reg2_Select, Timer_Select,
               IRQ,DataOut);
    input [7:0] DataIn;
    input [7:0] CountIn;
    input DRQ;
    input INTRQ;
    input FDC_Select, Timer_Select, Reg2_Select, Reg1_Select;

    output IRQ;
    reg IRQ;
    output [7:0] DataOut;
    reg [7:0] DataOut;

   always @(FDC_Select or Timer_Select or Reg2_Select or
            DataIn or CountIn or DRQ or INTRQ) // synthesis parallel_case
      begin
         if (FDC_Select)
            DataOut = DataIn;
        else if (Timer_Select)
            DataOut = CountIn;
        else if (Reg2_Select)
            DataOut =  8'b00000010; // Version
        else // Reg1_Select
            DataOut = {DRQ,INTRQ, 6'b000000};
      end

   // Emulate open drain signal for SS-30 bus
   always @(DRQ) begin
      if (~DRQ)
         IRQ = 1'b0;
      else
         IRQ = 1'bz;
   end
endmodule
```

WD2797 Interface

This module generates the active low WE, RE and CS for the WD2797as well as the Bi-directional buffer control signals. The Motor Run timer (MR_Timer) is pulsed whenever the WD2797 is accessed. The DDEN is an inverted SSO when in the default SWTPC mode. When in the 2 register mod it is controlled by a register bit.

```
module FDC_Ctrl(E,RW,FDC_Select,CTRL_Select,Mode_2Reg,DD_Ena,SSO,
                MR_Timer,WE,RE,CS_FDC,DDEN,FDC_OE,SS30_OE);

    input E, RW, FDC_Select, CTRL_Select, Mode_2Reg, DD_Ena, SSO;
   output MR_Timer, WE, RE, CS_FDC, DDEN, FDC_OE, SS30_OE;

   assign MR_Timer =  FDC_Select;

   assign DDEN     = (Mode_2Reg == 1'b0) ? ~DD_Ena : ~SSO;

   assign WE       = ~(E & ~RW & FDC_Select);
   assign RE       = ~(E &  RW & FDC_Select);
   assign CS_FDC   = ~FDC_Select;

   assign FDC_OE   =  FDC_Select  & ~RW;
   assign SS30_OE  = (FDC_Select | CTRL_Select) &  RW & E;

endmodule
```

Disk Drive Outputs

The motor starts when the FDC accessed. Reading or writing the addresses E018 to E10B (8018 to 801B) resets the timer. The motor runs for over 30 seconds and the timer is reset after each access.

The control register determines which floppy drive is selected. The select is only active if the motor is on or the DS_ENA bit is high. The Side_Sel bit is for double-sided drives.

The Drive0, Drive1, Drive2, Drive3 and Side_Select outputs are acitve low. The MR_Timer is active high and connects to pin 6 (MR) of the MC14541 timer.

New 3.5 inch drives are all hard wired to Drive 1 and need an IBM PC twisted cable as follows:

Pin	IBM PC	SWTPC and standard drives
10	Motor Enable A	Drive Select 0
12	Drive Select B	Drive Select 1
14	Drive Select A	Drive Select 2
16	Motor Enable B	Motor Enable all drives

Note: Drive A is after twist at the end of the cable.

```
 module DriveSelect(DS1, DS0, DS_INH, Motor, Mode_35Disk,
                   Drive3, Drive2, Drive1, Drive0, Motor_On);
    input DS1, DS0;
    input DS_INH, Motor, Mode_35Disk;
   output Drive3, Drive2, Drive1, Drive0;
   output Motor_On;
   reg Drive2, Drive0;

   always @(Mode_35Disk or Motor or DS1 or DS0 or DS_INH)
   // synthesis parallel_case
   begin
      if (Mode_35Disk == 1'b0) // Switch 3 ON - Logic Low
         begin
            Drive0 =  Motor;                         // Motor
            Drive2 = ~DS1 & ~DS0 & ~DS_INH & Motor; // Drive 0
         end
      else
         begin
            Drive0 = ~DS1 & ~DS0 & ~DS_INH & Motor; // Drive 0
            Drive2 =  DS1 & ~DS0 & ~DS_INH & Motor; // Drive 2
         end
   end

   assign Drive1 = ~DS1 &  DS0 & ~DS_INH & Motor;  // Drive 1

   assign Drive3 =  DS1 &  DS0 & ~DS_INH & Motor;  // Drive 3

   assign Motor_On = Motor;

endmodule
```

FDC Clock Circuit

The FDC clock is derived from a 12 MHz external oscillator. The standard 5.25 inch floppy requires 1.0 MHz, the 3.5 inch floppy drive requires 1.2 MHz and the 8 inch floppy drive requires 2.0 MHz. Osc20 is bit 3 (^h08) of Reg2 and Osc12 is bit 2 (^h04). Note: Xilinx App Note 104 recommends disconnecting all clocks during programming the CPLD

This is implemented by a programmable divider (3,5 or 6) followed by a divide by 2 to give a 50% duty cycle.

Osc12	Osc20	Divide by	Frequency
0	0	12	1.0 MHz
1	0	10	1.2 MHz
0	1	6	2.0 MHz
1	1	x	Don't Care

```
module ClockGen(Osc12,Osc20,Osc,Reset,FDC_Clk);
    input  Osc12, Osc20, Osc, Reset;
    output FDC_Clk;

    reg [2:0] Q; // programmable divider
    reg Q3;      // divide by 2

always@(posedge Osc or negedge Reset)
   begin
      if(~Reset)
         begin
            Q  = 3'b000;
            Q3 = 1'b0;
         end
      else if(Osc20 && Q == 3'b010) // Divide by 3 (0 to 2)
         begin
               Q  = 3'b000;
            Q3 = ~Q3;
         end
      else if(Osc12 && Q == 3'b100) // Divide by 5 (0 to 4)
         begin
               Q  = 3'b000;
            Q3 = ~Q3;
         end
      else if(~Osc12 && Q == 3'b101) // Divide by 6 (0 to 5)
         begin
               Q  = 3'b000;
            Q3 = ~Q3;
         end
      else
         Q = Q + 1;
   end

   assign FDC_Clk = Q3;

endmodule
```

Counter/Timer

The input signal, PulseIn, is synchronized to the 12 MHz clock. The new signal, Pulse, is the trigger for the state machine. When Pulse goes low the state machine advances to state Wait_2. This resets the count. When Pulse goes high the counting starts and runs until Pulse goes low. The value in the counter is held until there is a write to Timer_Select address.

```
module Timer(Osc,Timer_Select,RW,PulseIn,Reset,Count);
    input Osc, Timer_Select, RW, PulseIn, Reset;
    output [7:0] Count;
    reg [7:0] Count;

    reg Pulse;

    // State Register
    reg [2:0] current_state /* synthesis state_machine */;
    reg [2:0] next_state;
    // Define States
    parameter Wait_1   = 3'b000,
              Wait_2   = 3'b001,
              Counting = 3'b010,
              Done     = 3'b100;

   always @(Pulse or Timer_Select or RW or current_state)
   begin
      case(current_state)
      Wait_1:
         if(~Pulse)     next_state = Wait_2;
      Wait_2:
         if(Pulse)      next_state = Counting;
      Counting:
         if(~Pulse)     next_state = Done;
      Done:
         if(Timer_Select & ~RW) next_state = Wait_1;
      endcase
   end

   always @ (posedge Osc or negedge Reset)  // State Machine Registers
      begin
         if (~Reset)
            begin
               current_state = Wait_1;
              Pulse = 0;
            end
         else
            begin
               current_state = next_state;
              Pulse = PulseIn; // Sync to common clock
            end
      end

   always@(posedge Osc) // Counter Registers
      begin
         if(current_state == Wait_2)
            Count = 8'b0000000;
         else if(current_state == Counting)
            Count = Count + 1;
      end
endmodule
```