BURR-BROWN® APPLICATION BULLETIN
Mailing Address: PO Box 11400 • Tucson, AZ 85734 • Street Address: 6730 S. Tucson Blvd. • Tucson, AZ 85706
Tel: (602) 746-1111 • Twx: 910-952-111 • Telex: 066-6491 • FAX (602) 889-1510 • Immediate Product Info: (800) 548-6132

DDC101 EVALUATION FIXTURE PC INTERFACE BOARD

By Timothy V. Kalthoff

The DDC101 Evaluation Fixture is a modular design. With the use of the common DDC101 Evaluation Fixture PC Interface Board, different Device Under Test (DUT) Boards may be used. Different DUT boards can be used to evaluate single or up to 32 DDC101s in any package type. Design of the DUT boards and their use are detailed in the DDC101 Evaluation Fixture product data sheet.

This application note focuses on how to communicate directly to the PC Interface Board through a PC's printer port. This communication is used to configure the PC Interface Board as well as to configure and retrieve data from the DDC101s on the DUT Board. This will allow the user to write custom software to create user specific, extended capabilities not available in the standard program.

The software for the DDC101 Evaluation Board was written in Turbo Pascal®. The code can be readily modified to be used with other languages. All software examples listed use Turbo Pascal syntax.

DDC101 EVALUATION FIXTURE— PC INTERFACE BOARD

The DDC101 Evaluation Fixture's PC Interface Board is a data collection board designed to provide full operational control of the DDC101. The PC Interface Board provides control signals for the DDC101 and can collect up to 32,768 data words of DDC101 serial output. The PC Interface Board provides computer communication via the parallel interface port of an IBM compatible PC. The board can be used to evaluated up to 32 multiple DDC101s with their control signals connected in parallel and their outputs/inputs serially connected.

Control of the DDC101 from the PC is attained by loading control data into the PC Interface Board's registers. These registers are used for controlling system and data clock rates, integration time, number of DDC101's, readback delay, and the mode of operation. The following contains procedures for setting up the DDC101, collecting data from the DDC101 and writing to the on board RAM, and reading back the data from RAM.

PC INTERFACE BOARD SCHEMATICS

The PC Interface Board schematic is shown in Figure 1. The PC Interface Board uses two Xilinx Field Programmable Logic Arrays. The schematics for these are shown in Figures 2-8. For completeness, a single DDC101 DUT board schematic is shown in Figure 9.

LOCATING THE PC'S PARALLEL PORT'S ADDRESS

The PC interface board receives its instructions via the PC's parallel port. The address for the parallel printer port is found by reading it from the BIOS memory. LPT1, printer port 1's address, is located at hexidecimal address 40H, 08H (segment, offset). LPT2, printer port 2's address, is located at hexidecimal address 40H, 0AH.

ACCESSING THE PC INTERFACE BOARD REGISTERS

When information is sent to the PC Interface Board from the PC, bit 7 of the PC's Parallel Output Port is used to determine whether it is a register address or data to be written into a register. To write to a register, first send the register address with bit 7 low. This enables only the desired register to be written to. Then send the data with bit 7 high.

Note, whenever data is sent to the PC Interface Board (bit 7 high) the last register addressed will be written to. It is recommended that prior to transmitting each register's data, first send the register's address.

Table I lists the addresses of the registers which are contained in the PC Interface Board. Table II provides a description of these registers.

INITIALIZATION

The DDC101 PC Interface Board contains seven registers which must be initialized to establish control and timing for the DDC101 under test (refer to Table I and II). These registers control: Data Clock Rate, System Clock Rate, Integration Time, Data Transfer Delay, Number of DDC101s, DDC101 Setup Configuration, and PC Interface Board data collection.

DDC101 DATA COLLECTION MODE

Bit 5 of the Control Register determines the data transfer mode the DDC101 Evaluation Fixture will be in. The data is either being collected from the DDC101 under test and stored into the PC Interface Board's RAM or the data is being retrieved from the RAM and read back to the PC.

Data Storage—Setting bit 5 of the Control Register high initiates the data collect function of the PC Interface Board. At the end of each DDC101 conversion, Data Valid becomes active. Upon receipt of Data Valid, the PC Interface Board

Turbo Pascal®, Borland International, Inc.

Printed in U.S.A. November, 1994

initiates Data Transmit after delaying the value contained in the Read Delay Register. Data Transmit is active while 20 or 21 data clocks are sent per DDC101 (depends on the output data format). For each DDC101 data word received by the PC Interface Board, the serial data is fed into a serial to parallel shift register. Each output word is stored at a separate RAM address. After the output word is written to RAM, the RAM address counter is incremented.

The DDC101 PC Interface Board contains 32k by 24 bit RAM. This provides storage of a twenty-four bit word containing the DDC101's serial data in bits 20 to 0 (LSB = bit 0), output data format (unipolar or BTC) in bit 21, and overflow positive and negative in bits 23 and 22, respectively. Data is read back to the PC as six four bit nibbles.

Data Retrieval—Setting bit 5 of the Control Register low initiates the read data function of the PC Interface Board. Following a completed write cycle, the address counter will be put in a decrement mode and the RAM enabled for reading.

The data is read by the PC as six four bit nibbles. Writing to the desired Read Address outputs the specified nibble (refer to Table I). When all six nibbles have been read the memory is then decremented by writing to address $6F and the next data word can be read. Note that the data is inherently read last data word recorded first.

During the Read Mode, the PC Interface Board continues to issue FDS and Data Transmit commands to the DDC101. However, the data transmitted is not stored in RAM. The output of the parallel to serial shift register is disabled allowing the RAM data to be put on the bus for the read.

Setup Retrieval—Setting bit 1 of the Control Register low initiates the Setup Retrieval function of the PC Interface Board. The PC Interface Board instructs the DDC101(s) under test to send back their Setup Code information. This Setup Code information is stored on the PC Interface Boards RAM. Upon completion of the Setup Code Retrieval from the DDC101(s), the PC can then retrieve the Setup Code data from the PC Interface Board via setting control bit 5 low.

Setting control bit 1 low starts a machine in the PC Interface Board to retrieve the Setup Data from the DDC101(s) under test. The machine stores each DDC101's setup code at individual word locations of the PC Interface Board's RAM. The Setup Code is stored in the bottom three nibbles of each RAM word.

The Setup Data is read back from the PC Interface Board to the PC in a similar manner to the Data Retrieval operation, except only the bottom three nibbles need to be read.

SAMPLE PROGRAM

"Turbo Pascal Program Used for PC Interface Board Communications" lists a sample Turbo Pascal program which can be used with the PC Interface Board. This program may provide a useful starting point for a user customized application. This sample program is available on the DDC101 Evaluation Fixture's accompanying floppy diskette.

FUNCTION	ADDRESS	SHORT DESCRIPTION
Configuration A	resses:	
DCLKRATE	0 XXX 0000	Set Data Transfer Clock
SCLKRATE	0 XXX 0001	Set System Clock
FDS MSB FDS MID2 FDS MID1 FDS LSB	0 XXX 0010 0 XXX 0011 0 XXX 0100 0 XXX 0101	Set duration to Final Data Sample (Continuous integration, integration time)
SETUP MSB SETUP LSB	0 XXX 0110 0 XXX 0111	DDC101 Setup Configuration
RDLY MSB RDLY LSB	0 XXX 1000 0 XXX 1001	Post reset, system clock delay until data transfer
NUM DDCS	0 XXX 1010	Number of DDC101s
CTRL	0 XXX 1011	Control Register
DL SET	0 XXX 1100	Select bit transition to evaluate.

FUNCTION	ADDRESS	SHORT DESCRIPTION
Data Retrieval Mode, Read Functions:		
READ ADD0	0 000 XXXX	D3 - D0
READ ADD1	0 001 XXXX	D7 - D4
READ ADD2	0 010 XXXX	D11 - D8
READ ADD3	0 011 XXXX	D15 - D12
READ ADD4	0 100 XXXX	D19 - D16
READ ADD5	0 101 XXXX	D23 - D20
READ ADD6	0 110 XXXX	decrement memory
READ ADD7	0 111 XXXX	no op

X - Indicates Don't Care:

XXX - can be any value. Use of 111 will prevent selection of a Read Function. Selecting a Read Function during data collection mode will have no affect on the data.

XXXX - can be any value. However, use of 1111 will prevent selection of a Configuration Address. (Selection of a Configuration Address is not a problem unless data is subsequently written to this address).

TABLE I. Register Addresses.

REGISTER DESCRIPTIONS

SETDCLK—Loading with any other number will result in a Data Clock frequency according to the following formula.

Data Clock = (16MHz/2) / (N + 1)

For example, loading N = 1 results in a Data Clock frequency of 4MHz.

SETSCLK—Loading with any other number will result in a DDC101 System Clock frequency according to the following formula.

System = (16MHz/2) / (N + 1)

For example, loading N = 3 results in the DDC101's specified System Clock frequency of 2MHz.

FDS_MSB, FDS_MID2, FDS_MID1, FDS_LSB—Load these registers with the number of System Clock cycles between FDS pulses. For example, for a 1000 clock cycle integration time, load the binary value:

XXXX000 0000000 0000111 1101000

into FDS MSB, FDS MID2, FDS MID1, FDS LSB, respectively. Note, the first four bits of FDS MSB have no affect on the FDS timing.

SETUP_MSB, SETUP_LSB—Load the 6 LSBs of these registers with bits 11-6 and 5-0 of the DDC101's setup code. (Refer to product data sheet for setup code specifics). For example, for 16 Acquisition Clocks, 32 Samples/ Integration, 2 Integrations/Conversion, Unipolar Input Signal, and BTC Output Data Format, load the binary value:

X100101 X000101

into SETUP_MSB and SETUP_LSB, respectively. The CTRL register must be used to pass SETUP_MSB and SETUP_LSB's loaded data on to the DDC101.

RDLY_LSB, RDLY_MSB—Post DDC101 end of conversion reset, RDLY MSB and RDLY LSB control how many DDC101 System Clock cycles are delayed until the data transmission from the DDC101 is directed by the PC Interface Board (Read Delay clock cycles). For example, to wait 200 DDC101 System Clock cycles post reset, load the binary value:

XX00001 1001000

into RDLY_MSB and RDLY_LSB, respectively. Note, the first two bits of RDLY_MSB have no affect on read delay timing.

NUM_DDC101S—Load this register with the number of DDC101's that are cascaded. Up to 63 can be loaded. However, the 32k data words of RAM available on the board will limit the number of data points storable per DDC101. The DDC101 Evaluation Software limits this value at 32.

CTRL—This register controls the functions of the PC Interface Board as described:

BIT	FUNCTION	DESCRIPTION
6	Data Clock	1 = Data Clock provided externally. 0 = Use PC Interface Board Data Clock.
5	Data Mode	1 = Collect DDC101 data and store in RAM. 0 = Retrieve data from RAM to PC.
4	FDS Trig	1 = FDS Trigger provided externally. 0 = Use PC Interface Board FDS Trigger.
3	R Setup	1 = When strobed to a 1, releases DDC101 R_Setup and Transmits Setup Code stored in SETUP_MSB and SETUP_LSB registers. 0 = Sets DDC101 R_Setup Low (Active).
2	Test	1 = Sets DDC101 Test pin high. DDC101internal test current on. 0 = Sets DDC101 Test pin low. DDC101internal test current off.
1	Read Data/ Setup	1 = Output data retrieved from DDC101. 0 = Setup code retrieved from DDC101.
0	Reset Sys	1 = Sets DDC101 System Reset inactive (high). 0 = Sets DDC101 System Reset active (low).

DL SET—This register configures the DL Display Output. The DL display is operated by using the output of the PC Interface Board's DL Display as the Y-input to an oscilloscope, a ramp or triangle function used as input to the DDC101 is used as the oscilloscope's X-input.

Bits 5-2 determine the major carry to be evaluated.

1110	MSB
1101	MSB-1
•	•
•	•
0001	LSB+1
0000	LSB

Bits 1-0 determine the stairstep LSB load bits 0 and 1 as follows:

00 selects 20 Bit Level
01 selects 18 Bit Level
10 selects 16 Bit Level

TABLE II. Register Descriptions.

TURBO PASCAL PROGRAM USED FOR PC INTERFACE BOARD COMMUNICATIONS

```
{$A+,B–,E+,F+,G–,I+,L–,N+,O+,R+,S–,V–,X+}

program DDCDEMO;
{ Demonstration program to illustrate communications with the DDC101    }
{ PC Interface Board via an IBM compatible PC.                          }
{ * The setup configurations are programmed at the top of the main      }
{   program and can be easily altered by the user.                      }
{ * Program outputs are directed to the screen.  Simple program         }
{   modifications will allow the user to direct the outputs elsewhere.  }
{ * Note that the maximum number of data points retrievable are limited }
{   by Turbo Pascal's ability to allocate 64K bytes of data.  Using long }
{   integers to retrieve the data limits the number of points to 64K/4  }
{   or 16K bytes of data.  To retrieve the full 32K of data points that }
{   the PC Interface Board can collect requires the user to create a    }
{   large array function.                                               }
{                                                                       }
{                                          Tim Kalthoff                 }
{                                          Last revised August 2, 1994  }

uses Objects, Drivers, Views, Menus, App, Dos, CRT;

type
    RdWordPoint32= ^RdWordArray32;
    RdWordArray32= Array[1..32] of Word;

var
    XilinxCtrlCode, NumDDCCode,
    DataCkCode, SysCkCode, DLCode                  : Byte;
    DDCSetupCode, strbaddr, wraddr, readdr, I, J, K, Nt : Word;
    pPrintPort                                     : ^Word;
    DXmitDelayCode, IntCountCode                   : Longint;
    PCPort                                         : String[4];
    dataptr, oflowptr                              : pointer;
    dataTptr                                       : ^Longint;
    oflowTptr                                      : ^Byte;
    PDDCSetup                                      : RdWordPoint32;
    Error                                          : Boolean;
    DDCSetupText                                   : String[12];
    Itxt                                           : String[2];
    Ktxt                                           : String[4];
    dataTtxt                                       : String[7];


{ Procedure InitializeLPT defines the addresses for printer port        }
{ communication                                                         }
procedure InitializeLPT;
begin
    {Get LPT1 or LPT2 Address from BIOS Memory}
    if PCPort = 'LPT1' then pPrintPort := Ptr($40,$08);
    if PCPort = 'LPT2' then pPrintPort := Ptr($40,$0A);
    wraddr := pPrintPort^;                  {Parallel Port Write Address}
    readdr := wraddr + 1;                   {Parallel Port Read Address}
    strbaddr := wraddr + 2;          {Parallel Port Strobe Signal Address}
    Port[strbaddr] := 0;                          {Initialize Strobe High}
end;


{Procedure XilinxWrite sends data to address identified}
procedure XilinxWrite(XilinxAddr, XilinxData : byte);
begin
    Port[wraddr] := $7F AND XilinxAddr;     {Send Address}
    Delay(2);                               {Delay 2ms}
    Port[strbaddr] := 1;                    {Strobe Low}
    Delay(1);                               {Delay 1ms}
    Port[strbaddr] := 0;                    {Strobe High}
    Delay(1);                               {Delay 1ms}
    Port[wraddr] := $80 OR XilinxData;      {Send Data}
    Delay(2);                               {Delay 2ms}
    Port[strbaddr] := 1;                    {Strobe Low}
    Delay(1);                               {Delay 1ms}
    Port[strbaddr] := 0;                    {Strobe High}
    Delay(1);                               {Delay 1ms}
end;

{Procedure XilinxRefresh updates all control registers}
{on the PC Interface Board                             }
procedure XilinxRefresh;
begin
    Port[strbaddr] := 0;                    {Initialize port strobe high}
    Delay(1);

    {Setup Data Clock Rate}
    XilinxWrite(0, DataCkCode);             {Data Clock Register}

    {Setup System Clock Rate}
    XilinxWrite(1, SysCkCode);              {System Clock Register}

    {Send Integration Count Code}
    XilinxWrite(5,((IntCountCode AND $7F));
                                            {Integration Count LSB Register}
    XilinxWrite(4,((IntCountCode SHR 7)  AND $7F));
                                            {Integration Count LSB+1 Register}
    XilinxWrite(3,((IntCountCode SHR 14) AND $7F));
                                            {Integration Count MSB-1 Register}
    XilinxWrite(2,((IntCountCode SHR 21) AND $07));
                                            {Integration Count MSB Register}

    {Send DDC Setup Code}
    XilinxWrite(6, ((DDCSetupCode AND $FC0) SHR 6));
                                            {DDC Setup 6 MSB Reg}
    XilinxWrite(7, (DDCSetupCode AND $3F));
                                            {DDC Setup 6 LSB Reg}

    {Instruct Xilinx to Send Setup Code On To DDC}
    XilinxWrite(11, (XilinxCtrlCode AND $F7));
                                            {Xilinx Control Register}
    XilinxWrite(11, (XilinxCtrlCode OR $08));
                                            {Strobe DDC Setup Reset}

    {Send Data Transmit Delay Count Code}
    XilinxWrite(9, ((DXmitDelayCode AND $F80) SHR 7));
                                            {Data Xmit Count 5 MSB Register}
    XilinxWrite(8, (DXmitDelayCode AND $7F));
                                            {Data Xmit Count 7 LSB Register}

    {Send Add'l DDC Quantity (1 Assumed)}
    XilinxWrite(10, NumDDCCode);            {DDC Count Register}

    {Send DL Select}
    XilinxWrite(12, DLCode);                {DL Register}
end;


{Procedure getDataMem allocates memory for data.}
procedure getDataMem(Nsize: Word);
var
    memget  : Word;
    memerr  : boolean;

begin
    memerr := False;
    memget := Nsize*sizeof(Longint);
    if MaxAvail >= memget then getmem(dataptr, memget)
                                            else memerr := True;
    memget := Nsize*sizeof(Byte);
    if MaxAvail >= memget then getmem(oflowptr, memget)
                                            else memerr := True;
    if memerr = True then
    begin
        writeln('Non-Recoverable Memory Shortage—Program Halt');
        halt;
    end;
end;
```

```
{Procedure freeDataMem releases memory allocated for data.}
procedure freeDataMem(Nsize: Word);
var
    memfree :  Word;
    memerr  :  boolean;

begin
    memfree := Nsize*sizeof(Longint);
    freemem(dataptr, memfree);
    memfree := Nsize*sizeof(Byte);
    freemem(oflowptr, memfree);
end;


{Procedure ReadSetup reads back DDC101 setup data which is copied on}
{execution to PC Interface Board RAM.                               }
procedure ReadSetup(NumDDC, XilinxCtrlCode : Byte;
                    strbaddr, wraddr, readdr : Word;
                    var SURdError : Boolean;
                    var PDDCSetup : RdWordPoint32);
var
    FNum, J                          :  Longint;
    DFormat, BusyBit3                :  Integer;
    Nib, PortData, I                 :  Byte;
    iRd, K                           :  Word;

begin
    {Instruct Xilinx to fetch DDC101 setup to PC Interface Board RAM}
    XilinxCtrlCode := XilinxCtrlCode AND $FD; {Set R D/S Bit Low}
    XilinxWrite(11, XilinxCtrlCode);  {Xilinx Control Register}
    Delay(250);
    XilinxCtrlCode := XilinxCtrlCode AND $DD; {Set Wr/Rd Control Low}
    XilinxWrite(11, XilinxCtrlCode);  {Xilinx Control Register}

    {Wait For PC Interface Board Data Valid}
    K := 0;
    SURdError := False;
    BusyBit3 := (PORT[readdr] AND $08) SHR 3;   {Initialize bit 3}
    While (BusyBit3 <> 1) AND (K <= 20000) do
    begin
        BusyBit3 := (PORT[readdr] AND $08) SHR 3;
        inc(K);
        if (K mod 10) = 0 then delay(1);
        if K = 20000 then SURdError := True;
    end;

    if SURdError = False then
    begin
        for I := NumDDC downto 1 do
        begin
            PortData := $6F; {Decrement Xilinx RAM Memory Address}
            PORT[wraddr] := PortData;
            PORT[strbaddr] := 1;
            PORT[strbaddr] := 0;
            PORT[strbaddr] := 1;
            PORT[strbaddr] := 0;
            Fnum := 0;
            PortData := $3F;                {Nibble Pointer}
            for iRd := 1 to 3 do            {get 3 nibbles}
            begin
                PortData := PortData - $10;   {LSB+2,LSB+1,LSB nibbles}
                PORT[wraddr] := PortData;
                PORT[strbaddr] := 1;
                PORT[strbaddr] := 0;
                {Wait For PC Interface Board Data Valid}
                K := 0;
                BusyBit3 := (PORT[readdr] AND $08) SHR 3;{Initialize bit 3}
                While (BusyBit3 <> 1) AND (K <= 20000) do
                begin
                    BusyBit3 := (PORT[readdr] AND $08) SHR 3;
                    inc(K);
                    if (K mod 40) = 0 then delay(1);
                    if K = 20000 then
                    begin
                        SURdError := True;
                        J := 0;
                    end;
                end;
                nib := ((PORT[readdr] XOR $80) AND $F0) SHR 4;
                {XOR Inverts bit 7 (-busy), AND F0 gets rid of 4 lower bits,
                      Shift right puts the data bits in lower nibble}
                Fnum := Fnum SHL 4;
                FNum := Fnum OR nib;{Add nibble to final number}
            end;
            PDDCSetup^[I] := FNum;
        end;
    end;
    XilinxCtrlCode := XilinxCtrlCode OR $22; {Set Wr/Rd Control and}
                                             {R D/S Bits High       }
    XilinxWrite(11, XilinxCtrlCode);         {Xilinx Control Register }
    Delay(250);
end;


{Procedure ReadData reads back DDC101 conversion data stored in}
{PC Interface Board RAM.                                       }

procedure ReadData(dataptr, oflowptr : pointer; Nt : longint;
                   NumDDC, XilinxCtrlCode : Byte;
                   DDCSetupCode, strbaddr, wraddr, readdr : Word;
                   var RdError : Boolean);
{Needs 12 bit setup code from main program. This procedure checks data
valid until ready,  reads the data word 4 bits at a time, MSB first, reassembles
it, check for positive or negative overflow, determines data format, (unipolar or
BTC), returns a final 20 bit number (Fnum) in the proper format, then
decrements memory address.}

var
  DataT                          :  ^Longint;
  Oflow                          :  ^Byte;
  FNum, J                        :  Longint;
  DFormat, BusyBit3              :  Integer;
  Nib, PortData, I               :  Byte;
  iRd, K                         :  Word;

begin
    DataT := dataptr;                       {Initialize pointers}
    Oflow := oflowptr;
    Inc(DataT, Nt*NumDDC-1);  {Start pointers at end of memory allocated}
    Inc(Oflow, Nt*NumDDC-1);

    {Set Control Register to Readback Mode}
    XilinxCtrlCode := XilinxCtrlCode AND $DF;{Set Wr/Rd Control Bit Low}
    XilinxWrite(11, XilinxCtrlCode);        {Xilinx Control Register}

    {Wait For PC Interface Board Data Valid}
    K := 0;
    RdError := False;
    BusyBit3 := (PORT[readdr] AND $08) SHR 3;       {Initialize bit 3}
    While (BusyBit3 <> 1) AND (K <= 20000) do
    begin
        BusyBit3 := (PORT[readdr] AND $08) SHR 3;
        inc(K);
        if (K mod 10) = 0 then delay(1);
        if K = 20000 then RdError := True;
    end;
    {Get Data Stored in PC Interface Board RAM}
    if RdError = False then
    begin
        for J := Nt-1 downto 0 do           {For-Downto structures put }
        begin                               {last data at array bottom.}
            for I := NumDDC downto 1 do
            begin
                Fnum := 0;
                PortData := $6F;      {Prepare for RAM Memory Address}
                Port[wraddr] := PortData;{Decrement XilinxRAM
                                         Mem Address}
                Port[strbaddr] := 1;
                Port[strbaddr] := 0;
                Port[strbaddr] := 1;
                Port[strbaddr] := 0;
```

```
                {Prepare for RAM Memory Address    (PortData := $6F;)}
                for iRd := 1 to 6 do                        {Get 6 nibbles}
                begin
                    PortData := PortData - $10;
                                                {MSB, MSB-1,...,LSB nibble}
                    Port[wraddr] := PortData;
                    Port[strbaddr] := 1;
                    Port[strbaddr] := 0;
                    {Wait For PC Interface Board Data Valid}
                    BusyBit3 := (Port[readdr] AND $08) SHR 3; {init bit 3}
                    While (BusyBit3 <> 1) do
                        BusyBit3 := (Port[readdr] AND $08) SHR 3;

                    nib := ((Port[readdr] XOR $80) AND $F0) SHR 4;
                    {XOR Inverts bit 7 (-busy), AND F0 gets rid of 4 lower
                    bits, shift right puts the data bits in lower nibble}
                    if iRd = 1 then oflow^ := (nib AND $C) XOR $C;
                                                {bit4 = Oflow+ ; bit3 = Oflow-}
                                                {No overflow:       oflow^ = 0}
                                                {Positive overflow: oflow^ = 8}
                                                {Negative overflow: oflow^= 4}
                    Fnum := Fnum SHL 4;
                    FNum := Fnum OR nib;    {Add nibble to final number}
                end;

                {Check setup and bit 20 (BTC): 1=BTC, 0=Straight Binary:}
                if (DDCSetupCode AND $1 > 0) then
                begin                           {BTC Sign Extension}
                    if (Fnum AND $100000) > 0 then
                        Fnum := Fnum OR $FFF00000; {bits 31-22 := 1}
                    if (Fnum AND $100000) = 0 then
                        Fnum := Fnum AND $000FFFFF; {bits 31-22 := 0}
                end
                else                    {Straight Binary w/Offset}
                    Fnum :=(Fnum AND $000FFFFF) - $1000;
                                        {bits 31-22 := 0, subtract offset}
                {Check and adjust zero for No CDS with Bipolar Input:}
                if ((DDCSetupCode AND $C00)= 0) AND
                  ((DDCSetupCode AND $2)> 0)
                    then  Fnum :=Fnum - $80000;
                DataT^ := Fnum;
                dec(DataT);
                dec(Oflow);
            end;
        end;
    end;
    {Set Control Register to Data Write Mode}
    XilinxCtrlCode := XilinxCtrlCode OR $20; {Set Wr/Rd Control Bit High}
    XilinxWrite(11, XilinxCtrlCode);     {Xilinx Control Register}
end;


{Main Program  —  Uses above units to execute basic PC Interface Board }
{                 operations. Suitable for application specific modifactions. }
begin
    {Setup Printer Port}
    PCPort := 'LPT1';
    InitializeLPT;

    {Configure Control Registers}
    DataCkCode := 1;            {4MHz Data Clock}
          {DataCk = 16MHz/(2(SysCkCode+1) — 16MHz is the Xtal freq}
    SysCkCode := 3;             {2MHz System Clock}
          {SysCk = 16MHz/(2(SysCkCode+1) — 16MHz is the Xtal freq}
    IntCountCode := 1000;       {Integration Time=1000 Sys Clocks (500µs)}
    DDCSetupCode := $941;       {16 Acq, 32 Sam, 1 Int, Uni,BTC}
    DXmitDelayCode := 100;      {100 Delay Clocks}
    NumDDCCode := 1;            {1 DDC Under Test}
    XilinxCtrlCode := $2B;      {Internal Data Clock, Collect DDC101
                                 Data, Enable FDS, Reset Setup Off,
                                 DDC101 Test Signal Off, Read Data,
                                 Reset System Off}
    DLCode := $3A;              {MSB Major Carry, 16 Bit LSB}
    XilinxRefresh;

    Nt := 20;             {Retrieve 20 data points from PC Interface Board}
    Error := False;       {Set error flag}

    {Readback DDC101 Setup Code}
    ReadSetup(NumDDCCode, XilinxCtrlCode, strbaddr, wraddr, readdr,
              Error, PDDCSetup);
    ClrScr;
    if Error = True then
    begin
        writeln('Setup Readback Error.  Possible causes include: ');
        writeln('Power supply off, bad cable, incorrect PC port, ');
        writeln('or DDC101 not installed.       ');
    end
    else    {No readback error detected}
    begin
        for K := 1 to NumDDCCode do
        begin
            J := 1;
            DDCSetupText :='000000000000'; {Initialize text output variable}
            for I := 0 to 11 do
            begin
                if PDDCSetup^[K] AND J > 0 then
                    DDCSetupText[12-I] := '1'
                else
                    DDCSetupText[12-I] := '0';
                J := J SHL 1;
            end;
            writeln ('Setup DDC #',K,' = ', DDCSetupText);
        end;
    end;

    delay(5000); {5 second delay for data collection to PC interface board}

    getDataMem(Nt*NumDDCCode);
    {Read DDC101 Data Collected in PC Interface Board}
    ReadData(dataptr, oflowptr, Nt, NumDDCCode, XilinxCtrlCode,
            DDCSetupCode, strbaddr, wraddr, readdr, Error);
    writeln(' ');
    if Error = True then
    begin
        writeln('Data Retrieval Error.  Possible causes include:');
        writeln('Power supply off, bad cable,incorrect PC port, ');
        writeln('DDC101 not installed, or no external FDS trigger');
        writeln('(if so configured). ');
    end
    else    {No readback error detected}
    begin
        dataTptr := dataptr;
        oflowTptr := oflowptr;
        for K := 1 to Nt do
        begin
            for I := 1 to NumDDCCode do
            begin
                str(K:4,Ktxt);
                str(I:2,Itxt);
                str(dataTptr^:7,dataTtxt);
                writeln('Data Pt',Ktxt,'    DDC #',Itxt,' = ',dataTtxt,
                        ' , ',oflowTptr^);  {No overflow:       oflowTptr^  =  0}
                                            {Positive overflow: oflowTptr^  =  8}
                                            {Negative overflow: oflowTptr^ =  4}
                inc(dataTptr);
                inc(oflowTptr);
            end;
        end;
    end;
    freeDataMem(Nt*NumDDCCode);
end.
```

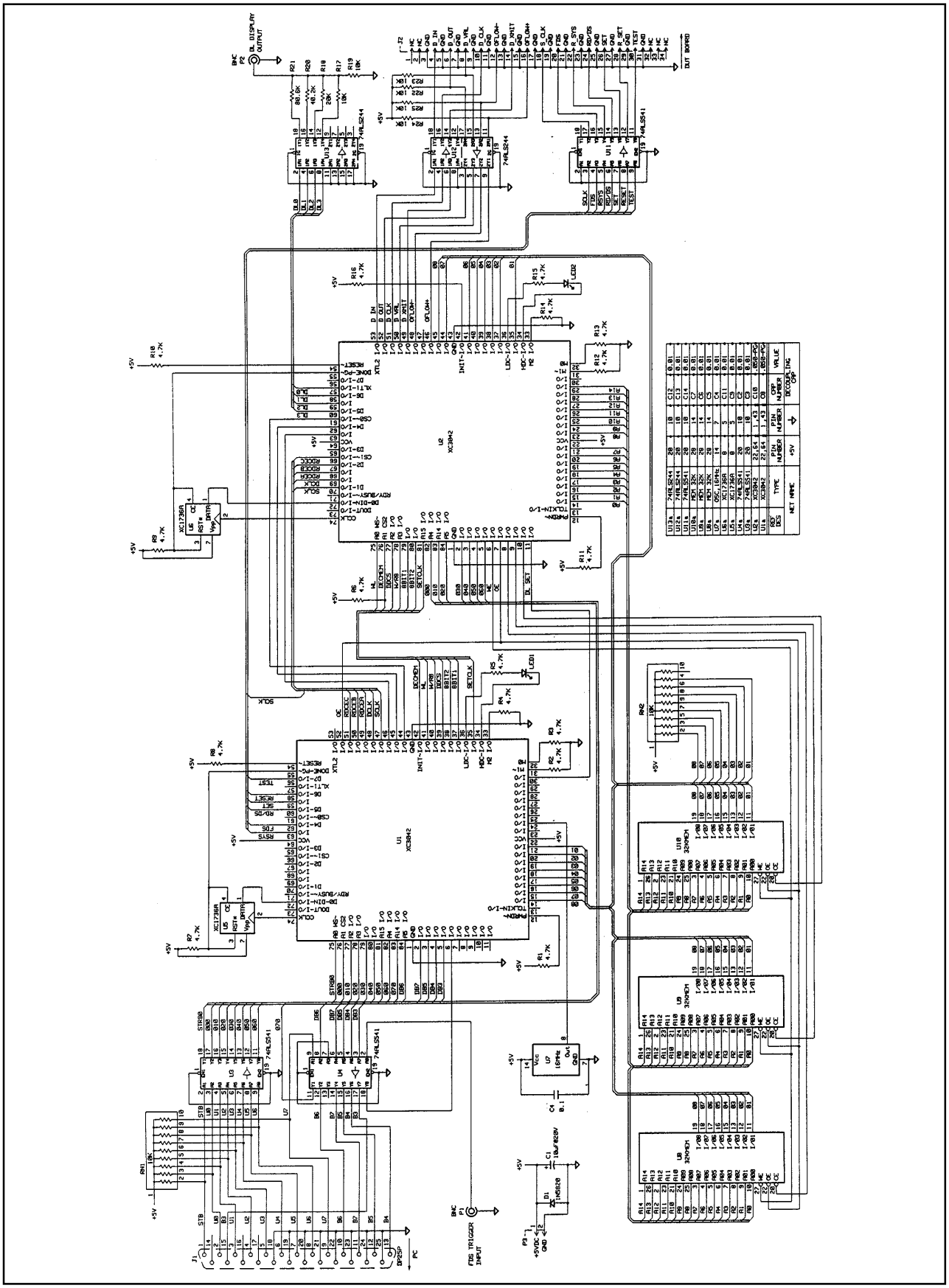

FIGURE 1. Circuit Diagram of PC Interface Board.

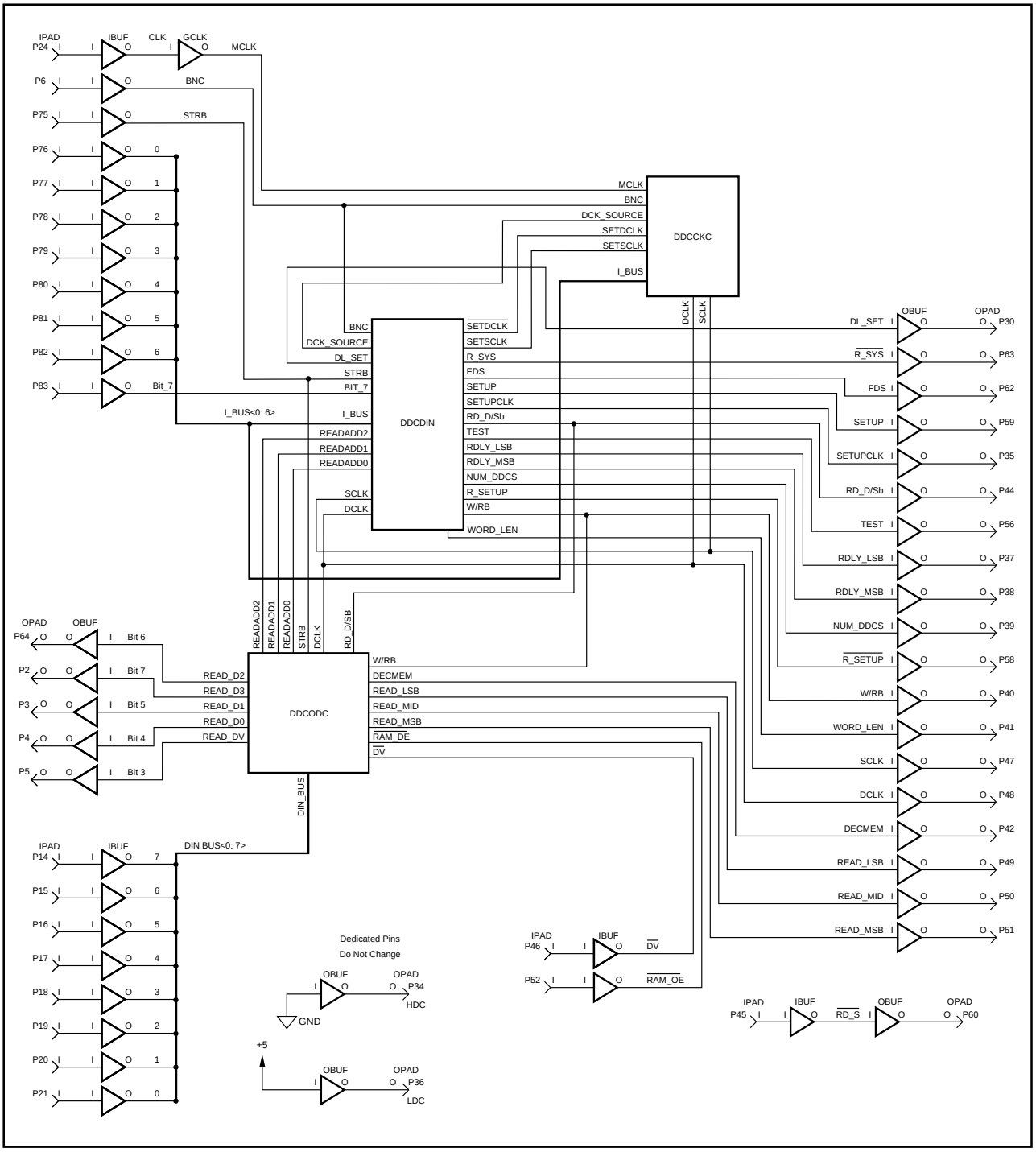

FIGURE 2. DDC101 PC Interface, XILINX U1 Setup PROM U5.

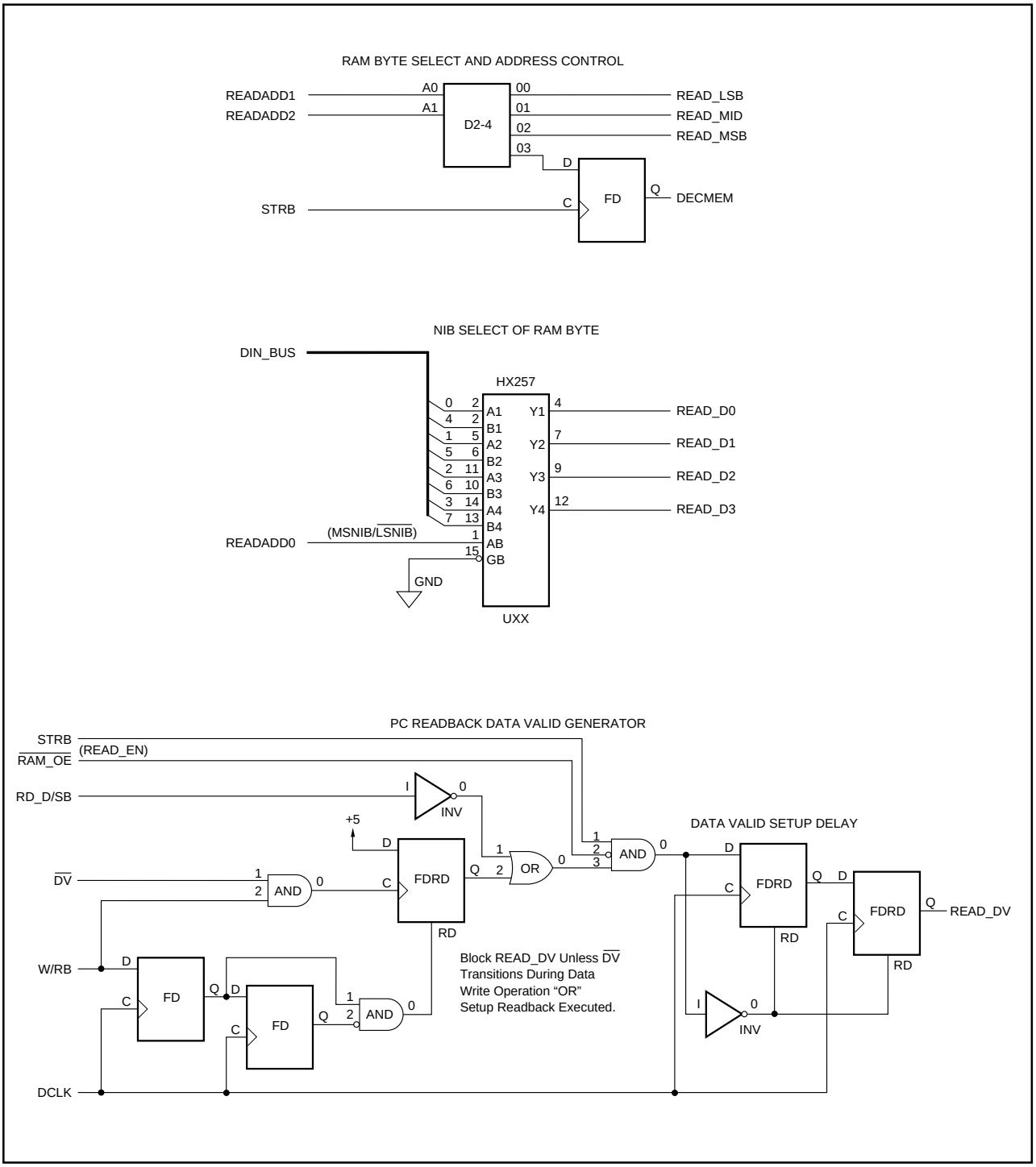

FIGURE 3. DDC101 Setup PROM U5, Output Data Control (DDCODC).

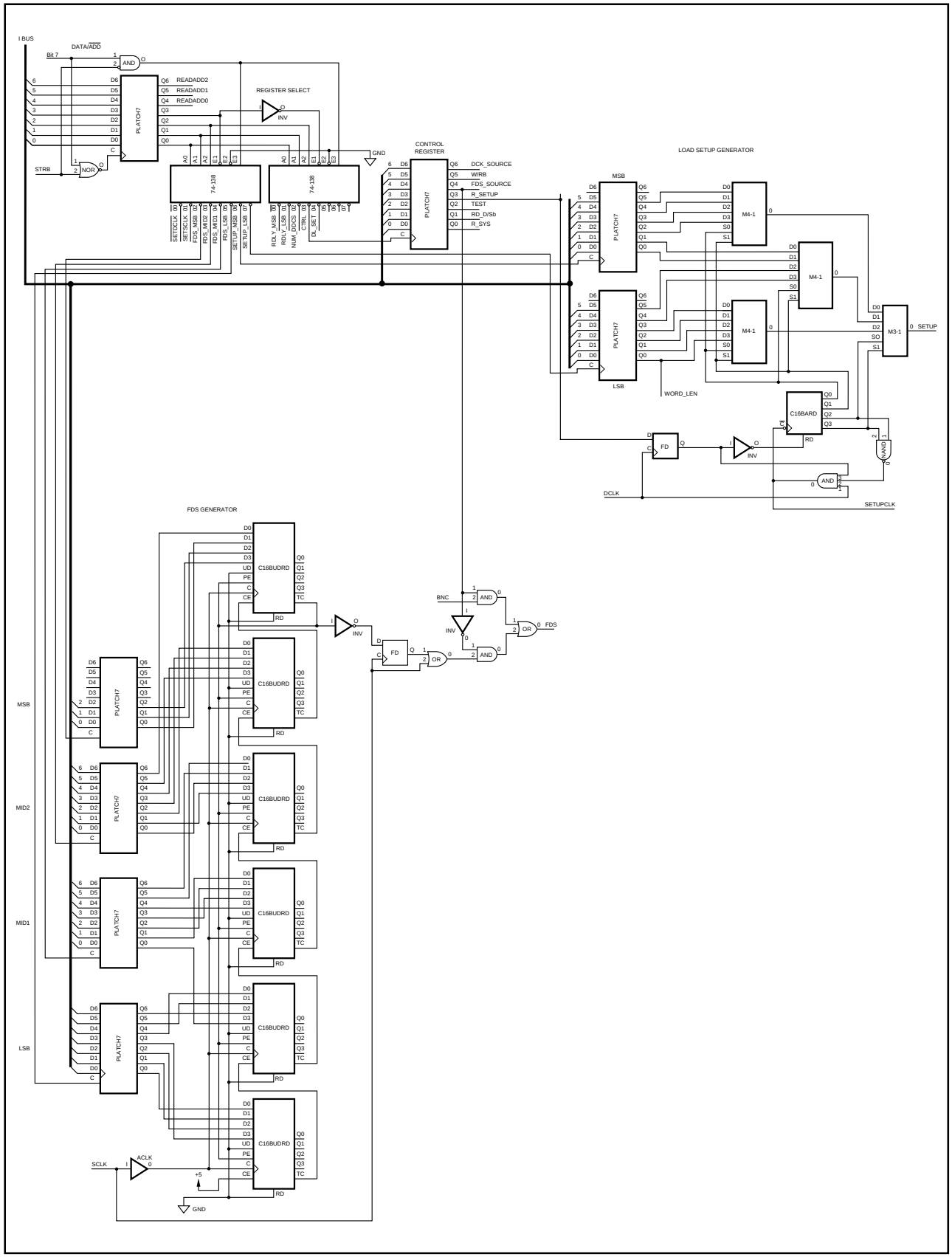

FIGURE 4. DDC101 Setup PROM U5, Data In (DDCDIN).

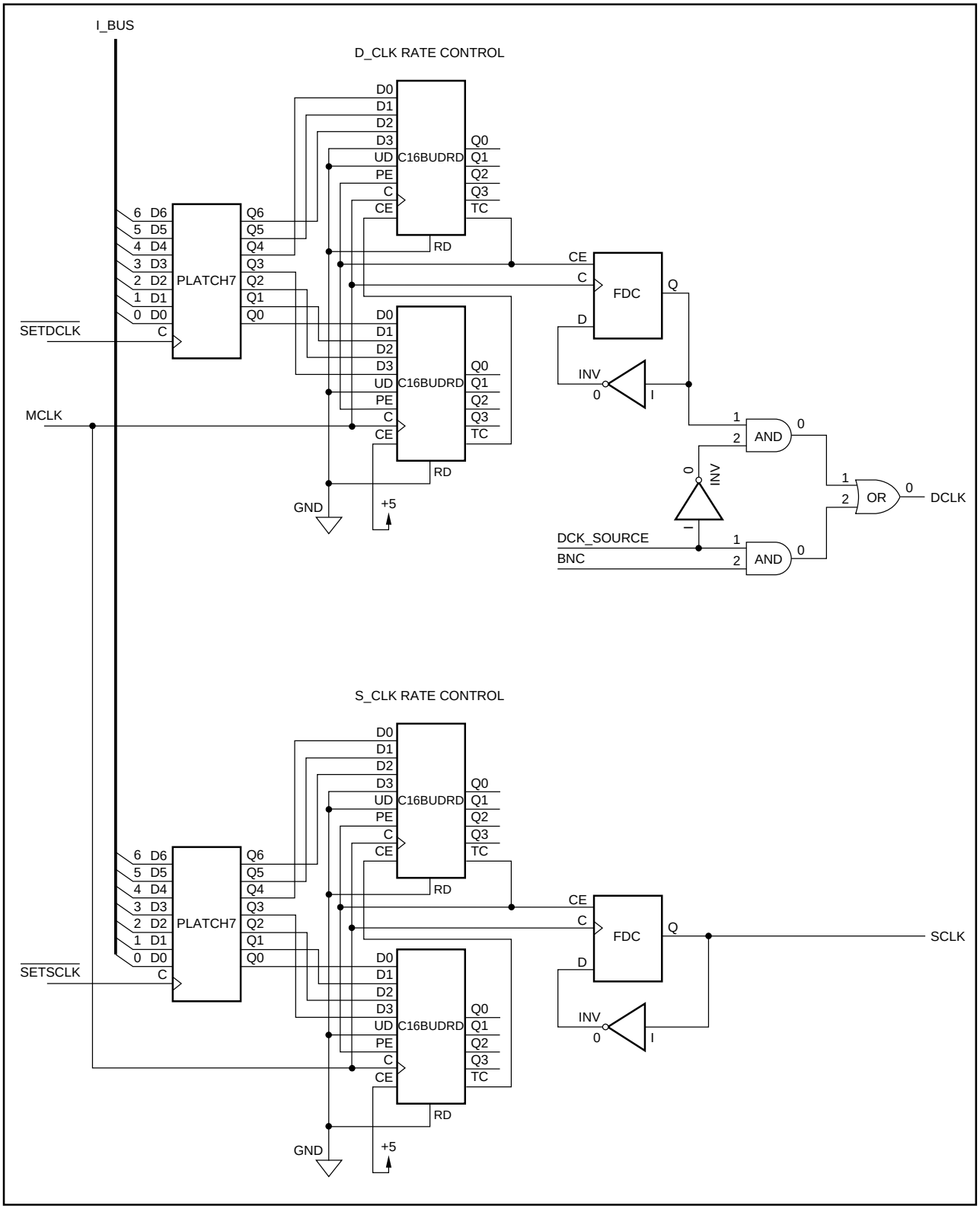

FIGURE 5. DDC101 Setup PROM U5, Clock Control (DDCCKC).

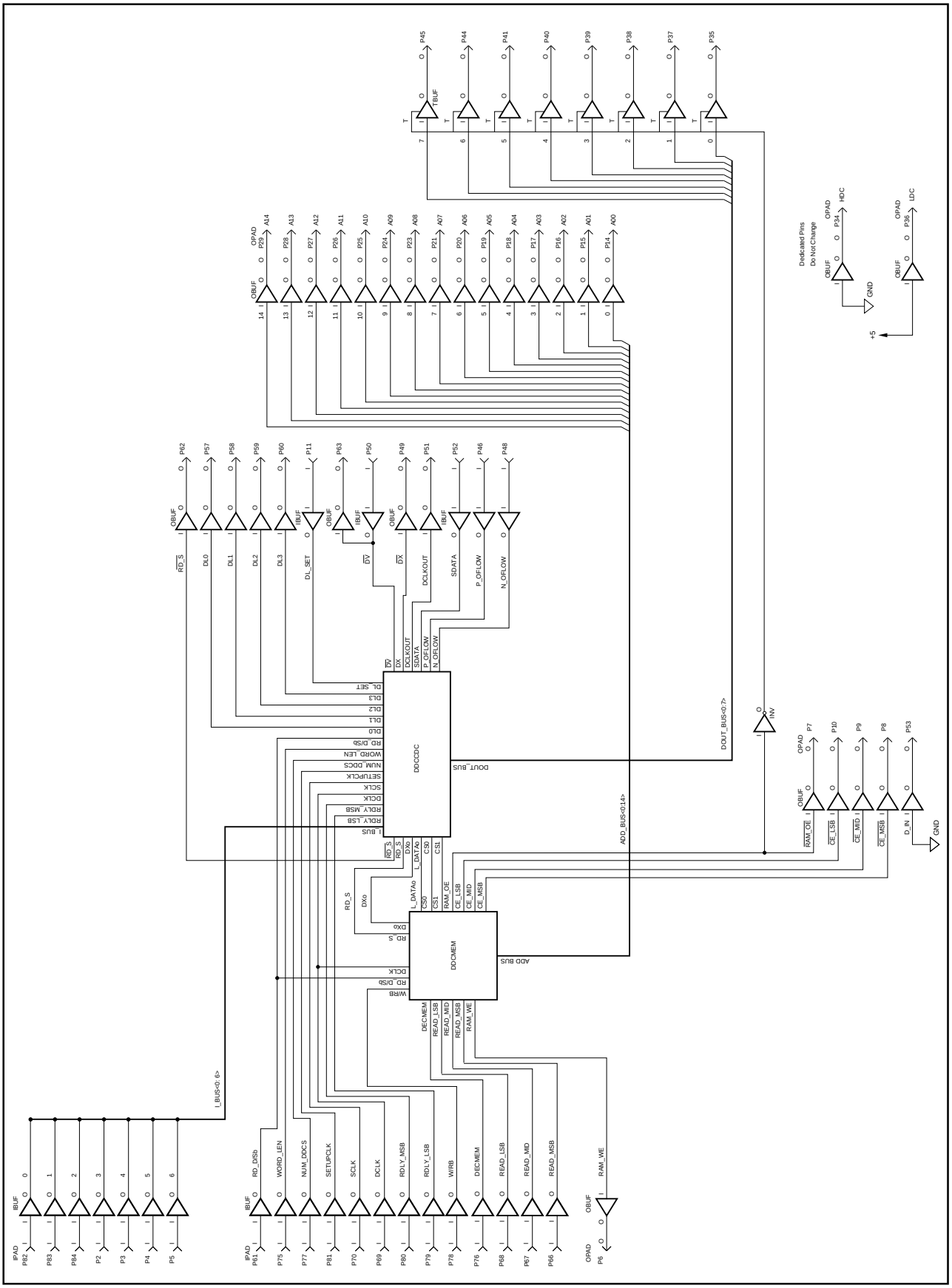

FIGURE 6. DDC101 PC Interface, XILINX U2 Setup PROM U6.

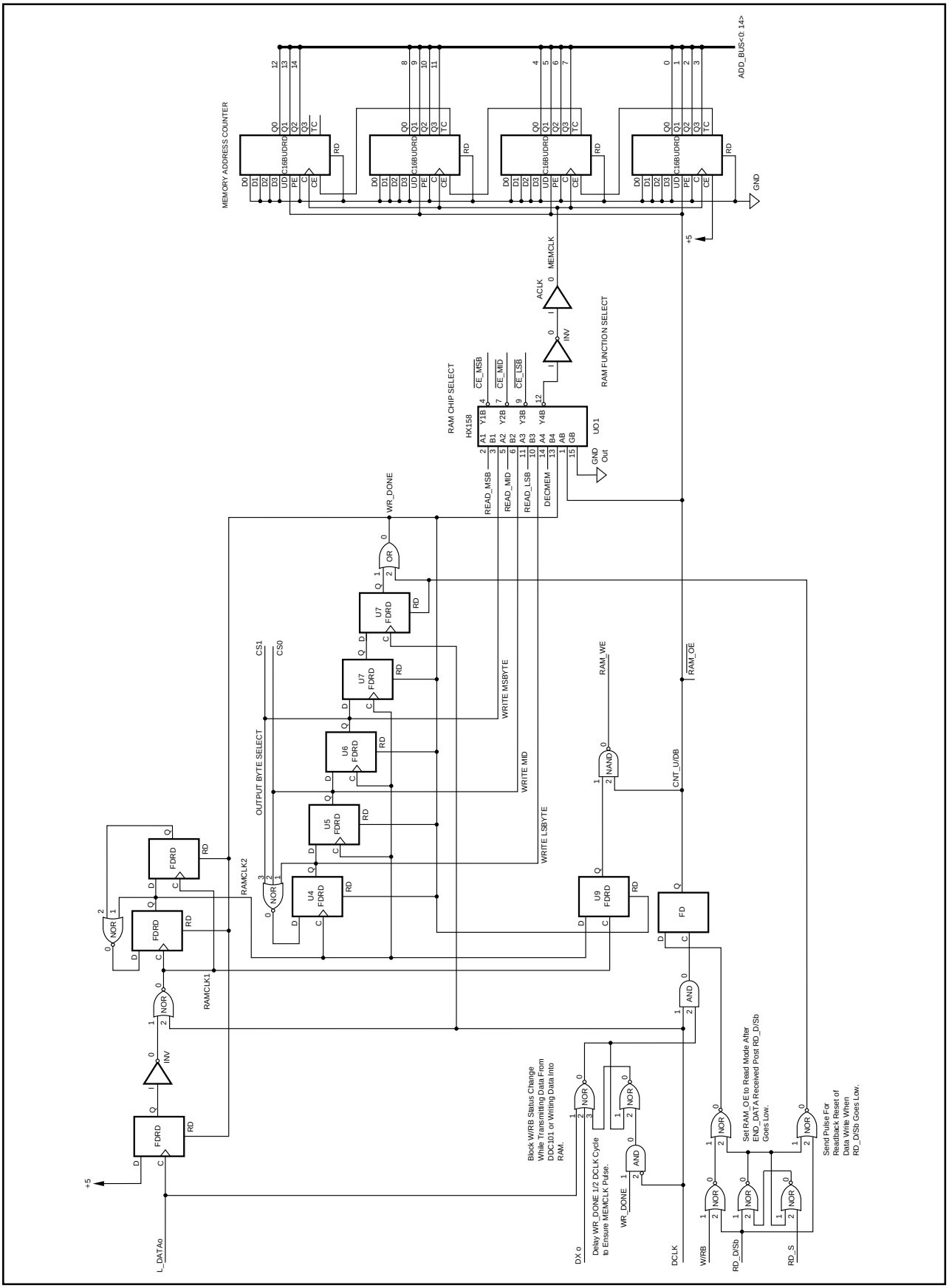

FIGURE 7. DDC101 Setup PROM U6, Memory Control (DDCMEM).

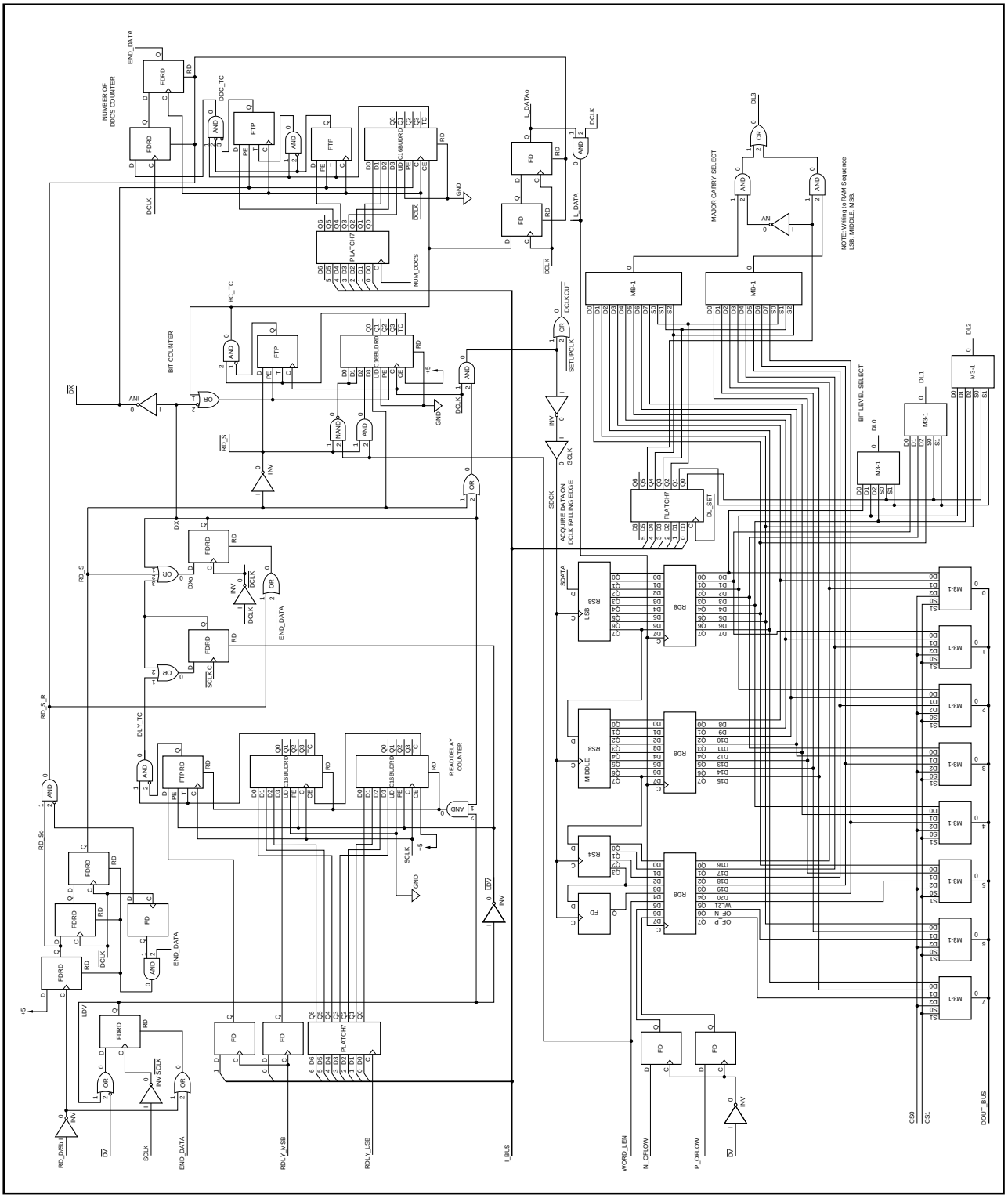

FIGURE 8. DDC101 Setup PROM U6, Collect Data Control (DDCCDC).

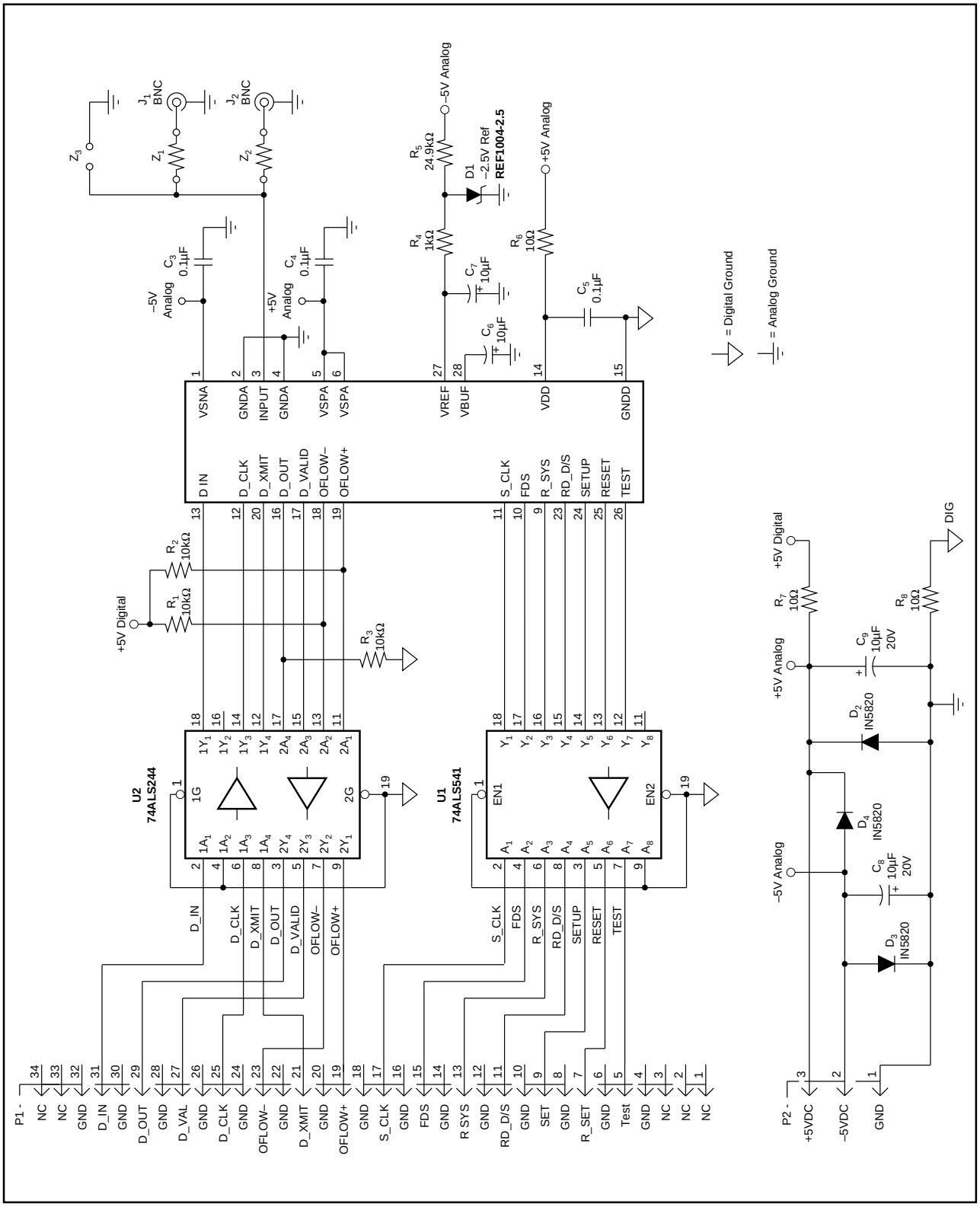

FIGURE 9. Circuit Diagram of DDC101-DUT Board.