IA3222/3223-EVB

IA3222/3223 Demo Board and CPLD Interposer User's Guide

Description

The IA3222/3223 Demo Board is a typical application circuit that exhibits all the features of the IA3222/3223 chipset. The chipset can be programmed by software to pass PTT certification worldwide. The integrated V.92 DAA offers an easy-to-use analog interface with an internal or external dc reference for seamless interfacing to a variety of systems. It allows easy building-block integration where audio codecs are either separate or integrated into DSPs. It is also ideal for non-modem systems requiring isolated DAAs, such as alarm systems, VoIP, and PBX FXO interfaces.

The CPLD interposer allows stand-alone evaluation of the chipset. This simple circuit takes the place of a microcontroller that would normally send off-hook commands and other settings to the IA3223's serial port. Instead of having to program a microcontroller, the user may control the IA3223's registers using manual switches.

Features

- Programmable worldwide telecom compliance with one hardware build
- V.92 (56 kbps) performance
- Virtually unlimited high-voltage insulation
- Highly-competitive BOM cost
- Lowest pin count (26) chipset
- High common-mode RF immunity without costly filtering
- Continuous dc and audio snooping with >5 MΩ Tip to Ring
- Parallel pick-up, line-in-use, ring, and "911" detection
- –86 dBm receiver noise floor
- +3 dBm transmit power
- Micropower line-side device powered from line
- 120 dB Caller ID common-mode rejection

1. IA3222/3223 Demo Board Photographs

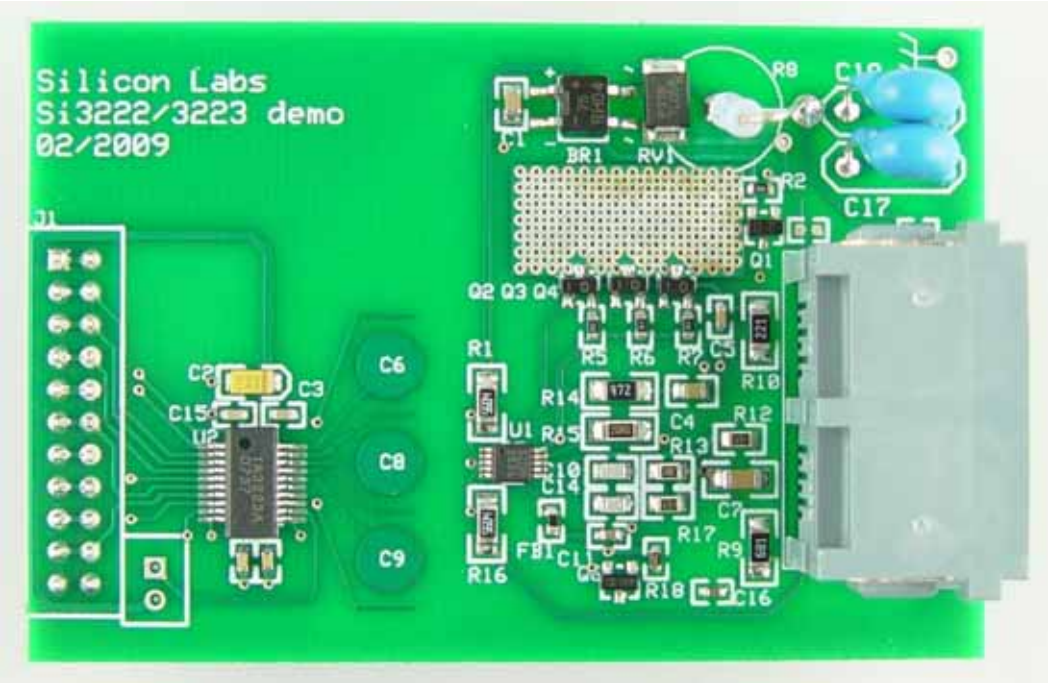

Figure 1. IA3222/3223 Demo Board

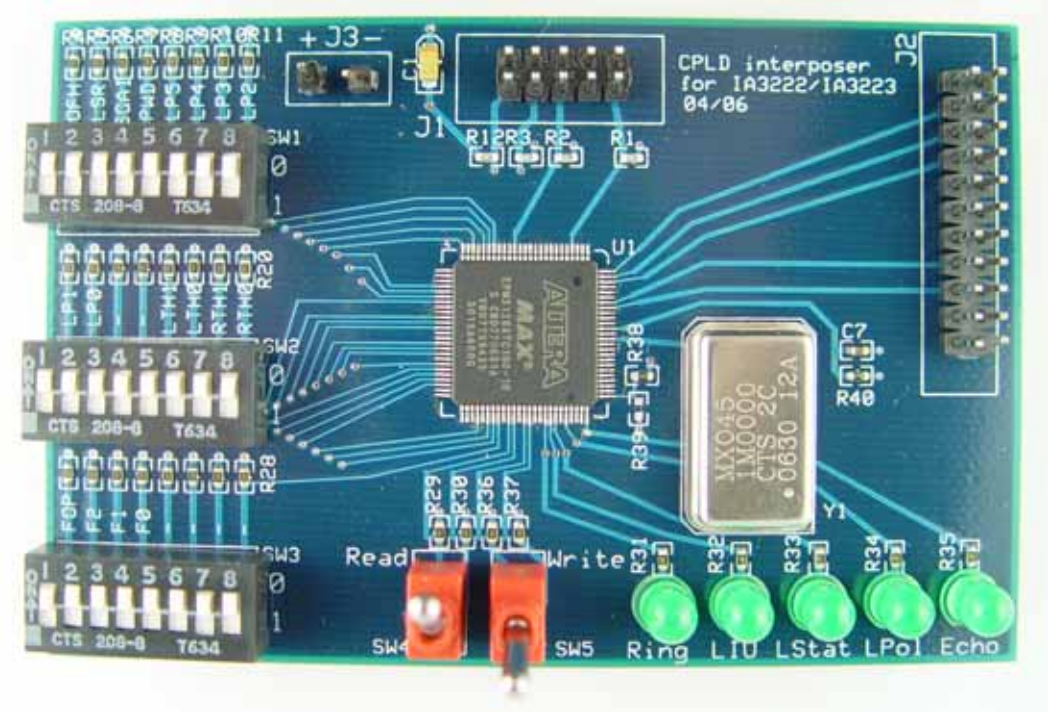

Figure 2. CPLD Interposer

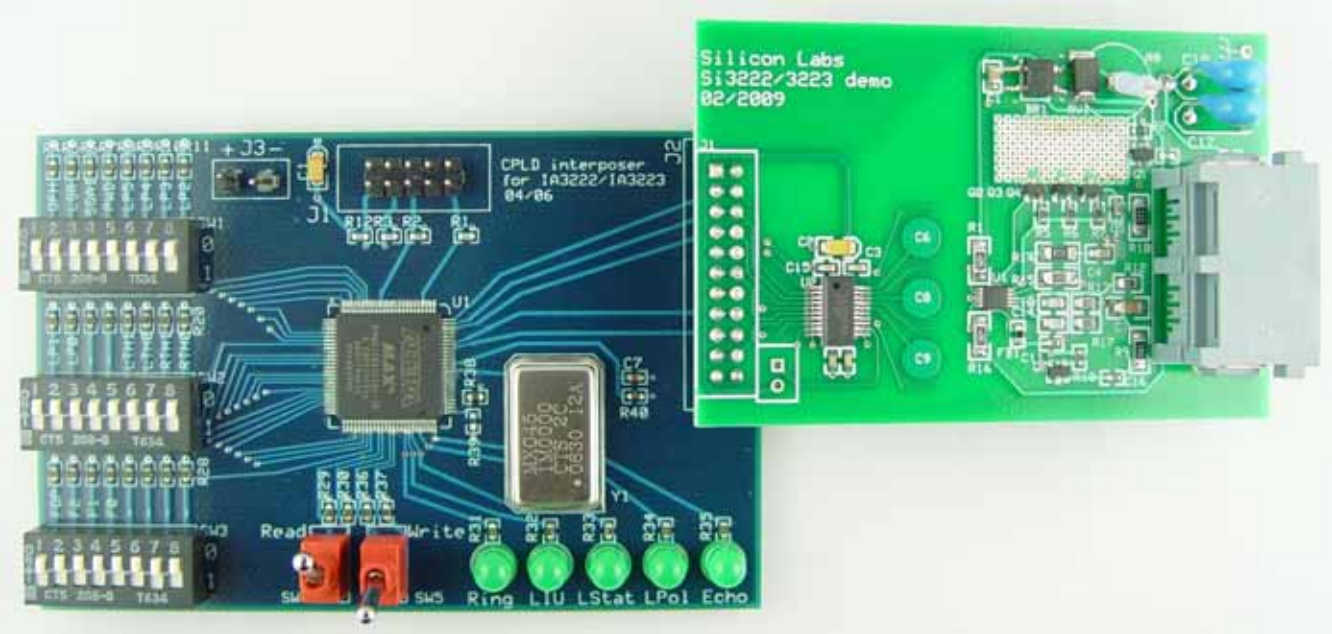

Figure 3. Demo Board and Interposer

2. Getting Started

The Interposer Board's simplified silkscreen is shown in Figure 4. Refer to the schematic for connector pinouts.

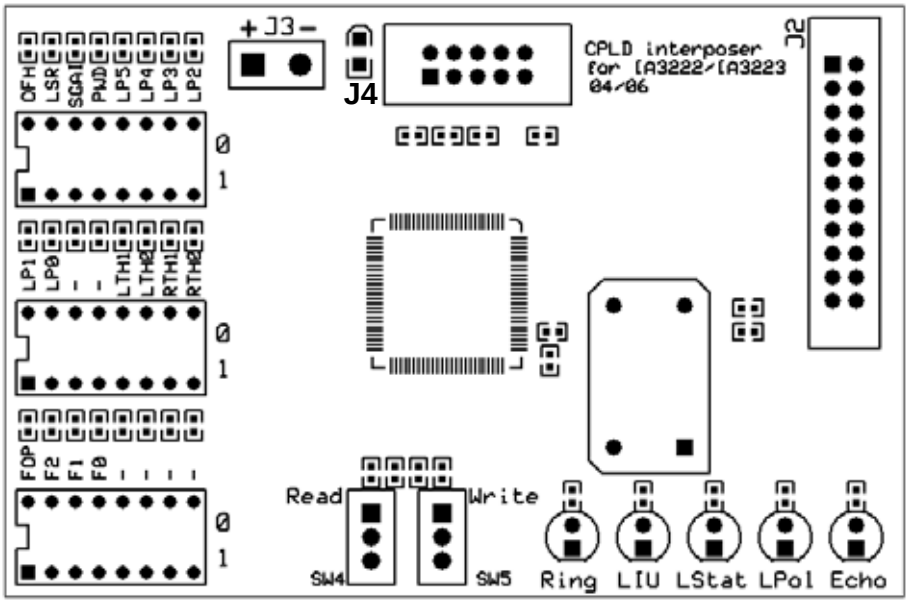

Figure 4. Interposer Board Simplified Silkscreen

The board must be powered using 3.3 V through J3 according to the polarity indicated on the silkscreen. The DAA Demo Board must be plugged into connector J2 on the Interposer as shown in Figure 3. The DAA Modem Side is powered via the Interposer.

Table 1 lists the DAA registers, which can be programmed via DIP switch banks on the Interposer Board. A complete description of these registers can be found in the IA3222/3223 data sheet.

Table 1. DAA Registers

A2	A1	A0	Register	D3	D2	D1	D0
0	0	0	Control	OFH	LSR	SGAIN	PWD
0	0	1	Line Side programming	LP5	LP4	LP3	LP2
0	1	0	Line Side programming	LP1	LP0	ECHO	REVID
0	1	1	Thresholds	LTH1	LTH0	RTH1	RTH0
1	0	0	Line status (read only)	RNG/PPU	LIU/LDN	LSTAT	LP
1	0	1	Dividers	Reserved	F2	F1	F0
1	1	X	Reserved	Reserved	Reserved	Reserved	Reserved

In order to program the DAA, the corresponding DIP switches must be set to the desired state; then, switch SW5 must be set to the momentary "write" position. Every push of SW5 makes the CPLD write all the DAA's read/write registers based on the current state of the DIP switches.

The line-status register and the ECHO bit in the Line-Side LSB register are read-only. If switch SW4 is set to "read," those read-only bits are monitored continuously and updated onto the LEDs. An LED turned on means a high state.

2.1. J1 Connector Pinout for the IA3222/3223 Demo Board

Table 2. J1 Connector Pin Descriptions*

Pin #	Name	Type	Description
1, 2	Extclk	Input	Optional external clock input for codec
3, 4	SClk	input	Serial data clock input from interposer board
5, 6	CS	Input	Chip select from interposer board
7, 8	SDIN	Input	Serial data input from interposer board
9, 10	VDD	Power	Power supply 3.3 V from interposer board
11, 12	TX	Input	Transmit analog input
13, 14	ACREF	Input/Output	Analog ac reference
15, 16	LINESTATUS	Output	Line status
17, 18	SDOUT	Output	Serial data output (register read)
19, 20	RX	Output	Receive analog output
21	GND	Power	Chassis ground
22	GND	Power	Signal and system ground
***Note:** See the IA3222/3223 data sheet for more information about differential or single-ended analog interfacing.			

The ExtClk input can be used in order to synchronize the IA3222/3223 internal codecs with an external codec. This helps prevent aliasing. Otherwise, the ExtClk input can be left open (there is an internal pull-down). The internal oscillator must then be selected. Refer to the data sheet for internal or external clock register settings.

2.1.1. Jumper Setting

A single jumper is available on the IA3222/3223 Demo Board. When it is open, the state of the hook switch is controlled by the serial port. When it is closed, the DAA goes off-hook regardless of the programmed hook-switch state. If the Interposer is used, the jumper is not needed to go off-hook.

2.2. Automatic Operation

In order to evaluate the IA3222/3223 DAA chipset as a replacement of a DAA product with a static hook control wire, the CPLD can recognize an active-high or active-low host command and produce the equivalent serial command to the IA3223. The host command signal is to be connected on the Interposer board at the net named tri_in, which is the common between R38 and R39. The leftmost switch of the SW1 bank (OFH) should be set to the polarity of the hook command that corresponds to the off-hook state. If the off-hook command is active high, OFH should be set to 1, and if the off-hook command is active low, OFH should be set to 0. At every transition of the hook command (tri_in signal), all the IA3223 registers are loaded based on the state of the DIP switches and of the hook command signal.

Manual operation is still possible with the hook command signal connected, but the meaning of the OFH DIP switch differs between automatic and manual operation. When a transition of tri_in is detected, OFH indicates the polarity of the off-hook command. When the write switch (SW5) is pushed momentarily, the state of OFH is copied to IA3223 register 0, where a high level means off-hook, and a low level means on-hook.

3. Demo Board Schematics

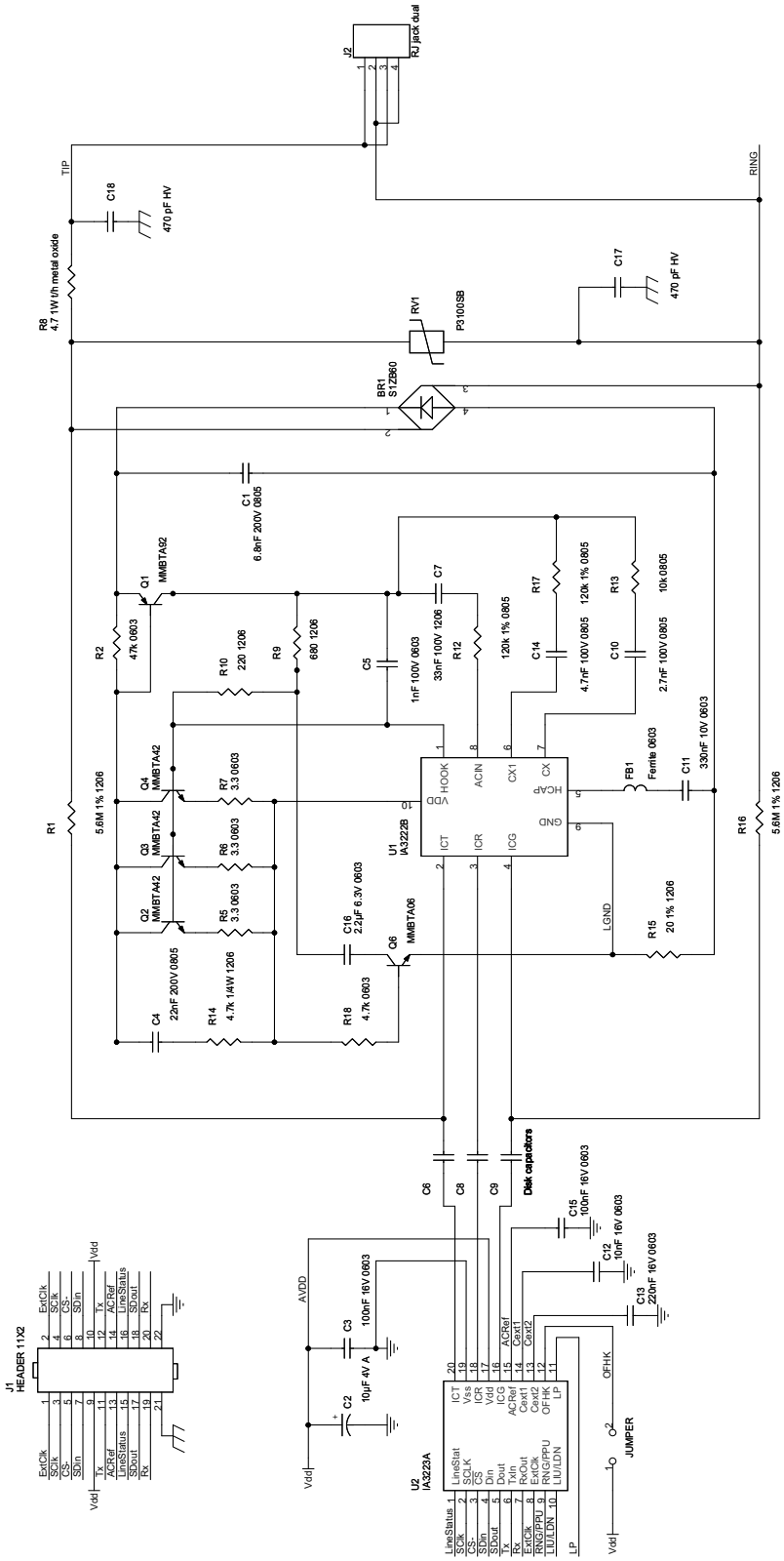

Figure 5. Demo Board Schematic

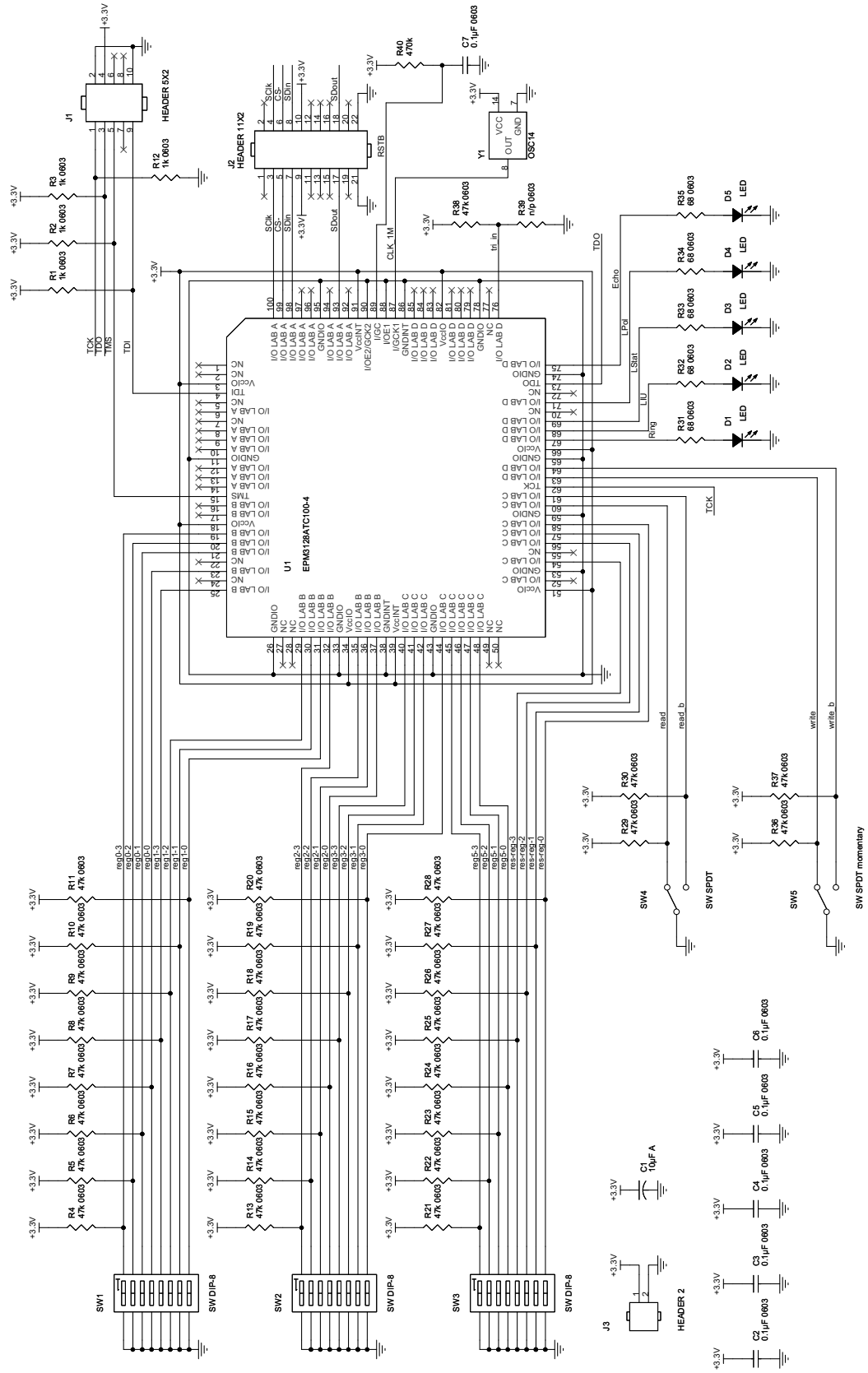

Figure 6. Interposer Board Schematic

4. Demo Board Bill of Materials

Table 3. Demo Board Bill of Materials

Quantity	Reference	Description
1	BR1	S1ZB60
1	C1	6.8 nF 200 V 0805
1	C2	10 µF 4 V A
2	C3, C15	100 nF 16 V 0603
1	C4	22 nF 200 V 0805
1	C5	1 nF 100 V 0603
1	C7	33 nF 100 V 1206
1	C10	2.7 nF 100 V 0805
1	C11	330 nF 10 V 0603
1	C12	10 nF 16 V 0603
1	C13	220 nF 16 V 0603
1	C14	4.7 nF 100 V 0805
1	C16	2.2 µF 6.3 V 0603
2	C17, C18	470 pF HV
1	FB1	Ferrite 0603 600Ω
1	Q1	MMBTA92
3	Q2, Q3, Q4	MMBTA42
1	Q6	MMBTA06
1	RV1	P3100SB
2	R1, R16	5.6 MΩ 1% 1206
1	R2	47 kΩ 0603
3	R5, R6, R7	3.3 Ω 0603
1	R8	4.7 Ω through hole metal oxide
1	R9	680 1206
1	R10	220 1206
2	R12, R17	120 kΩ 1% 0805
1	R13	10 kΩ 0805
1	R14	4.7 kΩ 1206
1	R15	20 Ω 1% 1206
1	R18	4.7 kΩ 0603
1	U1	IA3222B
1	U2	IA3223A

5. Interposer Bill of Materials

Table 4. Interposer Bill of Materials

Quantity	Reference	Description
1	C1	10 µF A
6	C2,C3,C4,C5,C6,C7	0.1 µF 0603
5	D1,D2,D3,D4,D5	LED green
4	R1,R2,R3,R12	1 kΩ 0603
29	R4–11, R13–30, R36–38	47 kΩ 0603
5	R31, R32, R33, R34, R35	68 0603
1	R39	N/P 0603
1	R40	470 kΩ
3	SW1, SW2, SW3	SW DIP-8
1	SW4	SW SPDT
1	SW5	SW SPDT momentary
1	U1	EPM3128ATC100-10
1	Y1	1 MHz DIP14 oscillator

6. Interposer Board CPLD Pin Assignment

```
Pin name   : Pin # : Type
---------------------------
reg_0[3]   : 19    : input
reg_0[2]   : 20    : input
reg_0[1]   : 21    : input
reg_0[0]   : 23    : input
reg_1[3]   : 25    : input
reg_1[2]   : 29    : input
reg_1[1]   : 30    : input
reg_1[0]   : 31    : input
reg_2[3]   : 32    : input
reg_2[2]   : 35    : input
reg_2[1]   : 36    : input
reg_2[0]   : 37    : input
reg_3[3]   : 40    : input
reg_3[2]   : 41    : input
reg_3[1]   : 42    : input
reg_3[0]   : 44    : input
reg_5[3]   : 45    : input
reg_5[2]   : 46    : input
reg_5[1]   : 47    : input
reg_5[0]   : 48    : input
read       : 60    : input
read_b     : 61    : input
write      : 63    : input
write_b    : 64    : input
ring       : 67    : output
liu        : 68    : output
lstat      : 69    : output
lp         : 71    : output
echo       : 75    : output
offhook    : 76    : input
clk1m      : 87    : input
rstb       : 89    : input
dout       : 93    : input
din        : 98    : output
csb        : 99    : output
sclk       : 100   : output
```

7. Interposer Board CPLD Verilog Code

```
/*******************************************/
   /*                                         */
   /*        weikang on 04/04/2006            */
   /*       modified on 06/12/2006            */
   /*                                         */
   /* sclk = 1MHz                             */
   /*                                         */
   /* simulator for p502 spi interface        */
   /*                                         */
   /* 1) change tri_in --> offhook            */
   /* 2) add offhook pin                      */
   /*    when offhook goes high/low-->reg0[3]=  */
   /*   (offhook&reg0[3]) | (~offhook&~reg0[3]) */
   /*    and generate write pulse             */
   /*                                         */
   /*                                         */
   /*                                         */
   /*******************************************/

`timescale 1ns/10ps

   module p502_spi_model_new_712(rstb, clk1m, dout,
                            reg_0, reg_1, reg_2, reg_3, reg_5,
                            write, write_b, read, read_b,
                            offhook,

                            echo, ring, liu, lstat, lp,
                            sclk, din, csb);

   /************** input and output  ***************/

   input         rstb;
   input         dout;
   input         clk1m;
   input [3:0]   reg_0, reg_1, reg_2, reg_3, reg_5;
   input         write, write_b;
   input         read, read_b;
   input         offhook;

   output        echo;
   output        ring, liu, lstat, lp;
   output        sclk, din, csb;

   reg           echo;
   reg           ring, liu, lstat, lp;
```

```
   /**************** p502_spi_model debouncing circuit **************/
// add on 07-12-06
   reg d0_offhook;
   reg d1_offhook;
   reg d2_offhook, d3_offhook, d4_offhook, d5_offhook;
   reg d6_offhook, d7_offhook, d8_offhook, d9_offhook;
   wire offhook_wr = (d0_offhook & ~d1_offhook) | (~d0_offhook & d1_offhook);

//   wire reg_0_sel = d2_offhook | d3_offhook | d4_offhook | d5_offhook |
//                    d6_offhook | d7_offhook | d8_offhook | d9_offhook;
   wire total_offhk = d2_offhook | d3_offhook | d4_offhook | d5_offhook |
                      d6_offhook | d7_offhook | d8_offhook | d9_offhook |
                      offhook;
   reg reg_0_sel;

   reg          sync_w, sync_w_b;
   reg          sync_r, sync_r_b;

   wire         q_w;
   wire         wrenb = ~(q_w & sync_w);
         assign q_w   = ~(sync_w_b & wrenb);
//   wire         wren  = ~ wrenb;
   wire         wren  = ~ wrenb | offhook_wr;

   wire         q_r;
   wire         rdenb = ~(q_r & sync_r);
         assign q_r   = ~(sync_r_b & rdenb);
   wire         rden  = ~ rdenb;

   reg          wk_wr;
   reg          clk;

   always @(posedge clk1m or negedge rstb)
     if (rstb==0) clk <= 1'b0;
     else         clk <= ~clk;

   always @(negedge clk or negedge rstb)
     if (rstb==0)
       begin
         sync_w   <= 0;
         sync_w_b <= 1;
         sync_r   <= 0;
         sync_r_b <= 1;
       end
     else
       begin
         sync_w   <= write;
         sync_w_b <= write_b;
```

```
        sync_r   <= read;
        sync_r_b <= read_b;
      end

   /*************** p502_spi_model register *************/

   reg   [7:0] read_cnt;
   reg   [2:0] bit_cnt;
   reg   [2:0] cnt, sync_cnt, s_sync_cnt;
   wire        t_sync_cnt = (sync_cnt==3'b101);
   reg         r_sync;
   reg         wr;
   reg         t_csb;
   wire        tri_in = ~rstb;
   wire        csb = (tri_in==1) ? 1'bZ : t_csb;
   reg         t_wk_read;
   reg         wk_read;
   reg         cnt_q0, cnt_q1, cnt_q2, cnt_q3;
   wire        clk_16 = cnt_q3;
   wire        clearb = rstb & ~(r_sync);
   wire        readen = (read_cnt==8'h02) | (read_cnt==8'h04);
   wire        tmp_csb = ~(wk_wr & ~clk_16) & ~(wk_read & ~clk_16 & readen);
   wire        bit_rstb = rstb & ~t_csb;
   reg   [7:0] data;
   wire  [7:0] r_data= (read_cnt==8'h02) ? 8'b10100000 : (read_cnt==8'h04) ?
8'b11000000 : 8'b0;
   wire  [2:0] downcnt = {~bit_cnt[2],~bit_cnt[1],~bit_cnt[0]};
   wire       tmp_din = (wk_wr==1) ? data[downcnt] : (wk_read==1) ? r_data[downcnt]
: 1'b0;
   reg         wk2_csb;
   wire        din = (tri_in==1) ? 1'bZ : tmp_din & ~wk2_csb & ~t_csb;
   wire        sclk = (tri_in==1) ? 1'bZ : (csb==1) ? 1'b1 : clk;
   wire        cnt_clearb = rstb & wk_read;

   /********************** add new function 07-12-2006 ********************/

   always @(posedge clk_16 or negedge rstb)
       if (rstb==0)
         begin
             d2_offhook <= 0;
             d3_offhook <= 0;
             d4_offhook <= 0;
             d5_offhook <= 0;
             d6_offhook <= 0;
             d7_offhook <= 0;
             d8_offhook <= 0;
             d9_offhook <= 0;
         end
```

```
        else
          begin
              d2_offhook <= offhook;
              d3_offhook <= d2_offhook;
              d4_offhook <= d3_offhook;
              d5_offhook <= d4_offhook;
              d6_offhook <= d5_offhook;
              d7_offhook <= d6_offhook;
              d8_offhook <= d7_offhook;
              d9_offhook <= d8_offhook;
          end


    always @(posedge clk or negedge rstb)
        if (rstb==0)
          begin
              d0_offhook <= 0;
              d1_offhook <= 0;
          end
        else
          begin
              d0_offhook <= offhook;
              d1_offhook <= d0_offhook;
          end

reg  w0_offhook, w1_offhook;
wire offhook_rstb = rstb & wrenb;
wire offhook_clk = (w0_offhook & ~w1_offhook) | (~w0_offhook & w1_offhook);

    always @(negedge clk or negedge rstb)
        if (rstb==0)
          begin
              w0_offhook <= 0;
              w1_offhook <= 0;
          end
        else
          begin
              w0_offhook <= total_offhk;
              w1_offhook <= w0_offhook;
          end

     always @(posedge offhook_clk or negedge offhook_rstb)
         if (offhook_rstb==0) reg_0_sel <= 0;
         else                 reg_0_sel <= ~reg_0_sel;

    /*************** *************/

     always @(posedge clk or negedge rstb)
```

```
        if (rstb==0)   cnt_q0 <= 0;
        else           cnt_q0 <= ~cnt_q0;

    always @(negedge cnt_q0 or negedge rstb)
        if (rstb==0)   cnt_q1 <= 0;
        else           cnt_q1 <= ~cnt_q1;

    always @(negedge cnt_q1 or negedge rstb)
        if (rstb==0)   cnt_q2 <= 0;
        else           cnt_q2 <= ~cnt_q2;

    always @(negedge cnt_q2 or negedge rstb)
        if (rstb==0)   cnt_q3 <= 0;
        else           cnt_q3 <= ~cnt_q3;

    always @(posedge wren or negedge clearb)
        if (clearb==0) wr <= 1'b0;
        else           wr <= 1'b1;

    always @(posedge clk_16 or negedge clearb)
        if (clearb==0) wk_wr <= 1'b0;
        else           wk_wr <= wr;

     always @(posedge clk_16 or negedge clearb)
        if (clearb==0)     cnt <= 3'b0;
        else if (wk_wr==1) cnt <= cnt + 1;

     always @(negedge clk1m or negedge rstb)
        if (rstb==0)  s_sync_cnt <= 3'b0;
        else          s_sync_cnt <= cnt;

     always @(negedge clk or negedge rstb)
        if (rstb==0)  sync_cnt <= 3'b0;
        else          sync_cnt <= s_sync_cnt;

     always @(negedge clk or negedge rstb)
        if (rstb==0)  r_sync <= 1'b0;
        else          r_sync <= t_sync_cnt;

     always @(negedge clk or negedge bit_rstb)
        if (bit_rstb==0)            bit_cnt <= 3'b111;
        else if (t_csb==0 & (wk_wr==1 | wk_read==1)) bit_cnt <= bit_cnt + 1;

// add on 07-12-06
     wire       t_offhook = (reg_0[3] & offhook) | (~reg_0[3] & ~offhook);
     wire [3:0] t_reg_0 = (reg_0_sel==1) ? {t_offhook,reg_0[2:0]} : reg_0;
     always @(t_reg_0 or reg_1 or reg_2 or reg_3 or reg_5 or wk_wr or sync_cnt)
        case (sync_cnt)
```

```
      3'b000 : data = {4'b0000,t_reg_0};
      3'b001 : data = {4'b0001,reg_1};
      3'b010 : data = {4'b0010,reg_2};
      3'b011 : data = {4'b0011,reg_3};
      3'b100 : data = {4'b0101,reg_5};
      default: data = 8'b00000000;
     endcase

always @(posedge clk_16 or negedge rstb)
    if (rstb==0) t_wk_read <= 1'b0;
    else         t_wk_read <= rden;

always @(negedge clk or negedge rstb)
    if (rstb==0) wk_read <= 1'b0;
    else         wk_read <= t_wk_read;

always @(posedge clk_16 or negedge cnt_clearb)
    if (cnt_clearb==0) read_cnt <= 8'b0;
    else               read_cnt <= read_cnt + 1;

always @(negedge clk1m or negedge rstb)
    if (rstb==0) t_csb <= 1;
    else         t_csb <= tmp_csb;

always @(negedge clk1m or negedge rstb)
    if (rstb==0) wk2_csb <= 1;
    else         wk2_csb <=t_csb;

always @(posedge clk or negedge rstb)
   begin
    if (rstb==0)
     begin
       echo <=0;
       ring <=0;
       liu  <=0;
       lstat<=0;
       lp   <=0;
     end
    else if (wk_read==1 & t_csb==0)
           begin
              if (read_cnt==8'h02 & bit_cnt==3'b110)
                  echo <= dout;
              else if (read_cnt==8'h04 & bit_cnt==3'b100)
                        ring <= dout;
                   else if (read_cnt==8'h04 & bit_cnt==3'b101)
                             liu <= dout;
                        else if (read_cnt==8'h04 & bit_cnt==3'b110)
                                  lstat <= dout;
```

```
                                else if (read_cnt==8'h04 & bit_cnt==3'b111)
                                        lp <= dout;
            end
      end
 /********************** add new function ********************/

reg offhk;

always @(posedge wk_wr or negedge rstb)
  if (rstb==0) offhk <= 0;
  else         offhk <= t_reg_0[3];


 /********************** call modules ********************/


 endmodule
```

CONTACT INFORMATION

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
Tel: 1+(512) 416-8500
Fax: 1+(512) 416-9669
Toll Free: 1+(877) 444-3032

Please visit the Silicon Labs Technical Support web page:
https://www.silabs.com/support/pages/contacttechnicalsupport.aspx
and register to submit a technical support request.