Using 8255 Mode 1 and Mode 2 with PCITG Card

Vincent Yang

Introduction

8255 is a fundamental IO chip. It's a good target for practicing system integration. PCITG is a CPLD based PCI target device equip with two 8255 chips, and more, we route $INTR_A$ and $INTR_B$ to PCI Interrupt, which enhance the flexibility of application. And here, we will demonstrate using one 8255 with Port A in mode 2 and Port B in mode 1 to construct a VME-like A15:D8 conversion.

When 8255 operate at mode 1 or mode 2, we need to send some handshaking signals through Port C. So, we have to build an external logic. All the schematic drawing, VHDL code and software code are listed in the following page. Have fun.

Signal Description

8255 Side

Control Word: 0xC4

Port A: Mode 2, Bi-Directional

Port B: Mode 1, Output

Port C: Control Signals

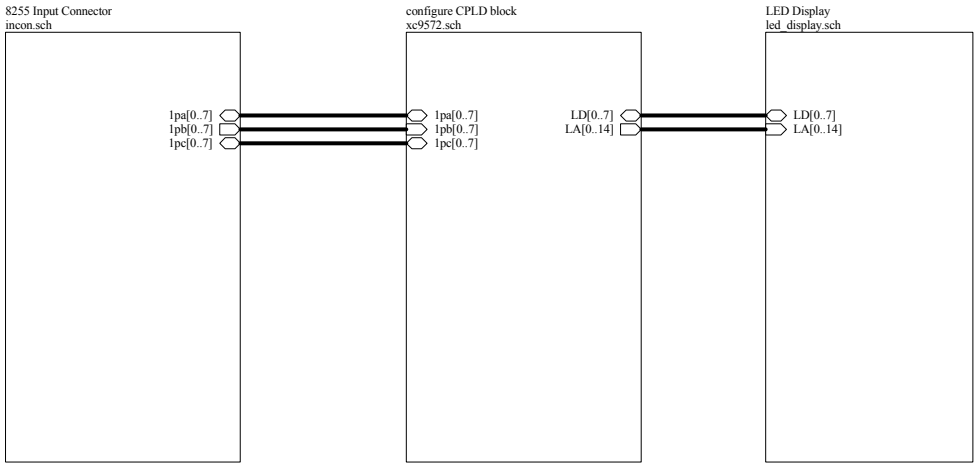
Signals	In/Out	Usage
PA[7..0]	Bi-Direction	Data
PB[6..0]	Output	Address
PB7	Output	Read/Write
PC0	Output	$INTR_B$
PC1	Output	OBF_B
PC2	Input	ACK_B
PC3	Output	$INTR_A$
PC4	Input	STB_A
PC5	Output	IBF_A
PC6	Input	ACK_A
PC7	Output	OBF_A

On VME Side

Signals	In/Out	Usage
LD[7..0]	Bi-Direction	Data
LA[14..0]	Output	Address
RW	Output	Read/Write
DS	Output	$INTR_B$

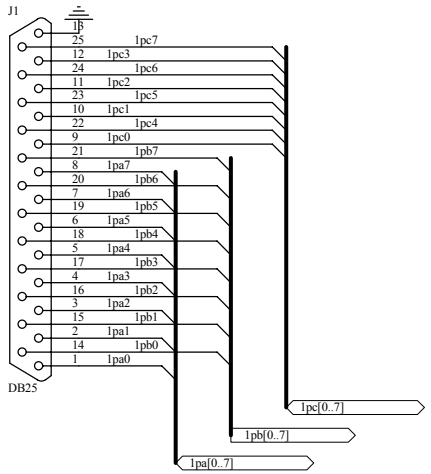
PCITG 8255 MODE 1 & 2 TEST PCB

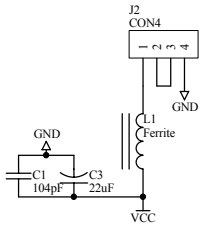
Revision			
Rev	Description	Date	Approved
1.00	Initial Draft		Vincent

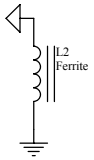


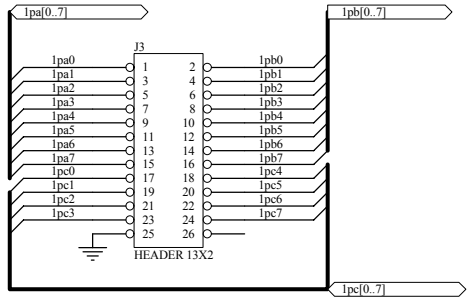
Title		Revision:
PCITG 8255 MODE 1 & 2 TEST PCB		
Size: B	Number:	
Date: 15-Apr-2004	Time: 16:17:11	Sheet1 of 4
File: C8255T.prj		

SOLITON Technologies CO., LTD.
Power by Soliton Technologies Co., LTD. Tel : 03-6566996
2F, No.198, Guangming 3rd Road, Jhubei City, Hsinchu, ROC





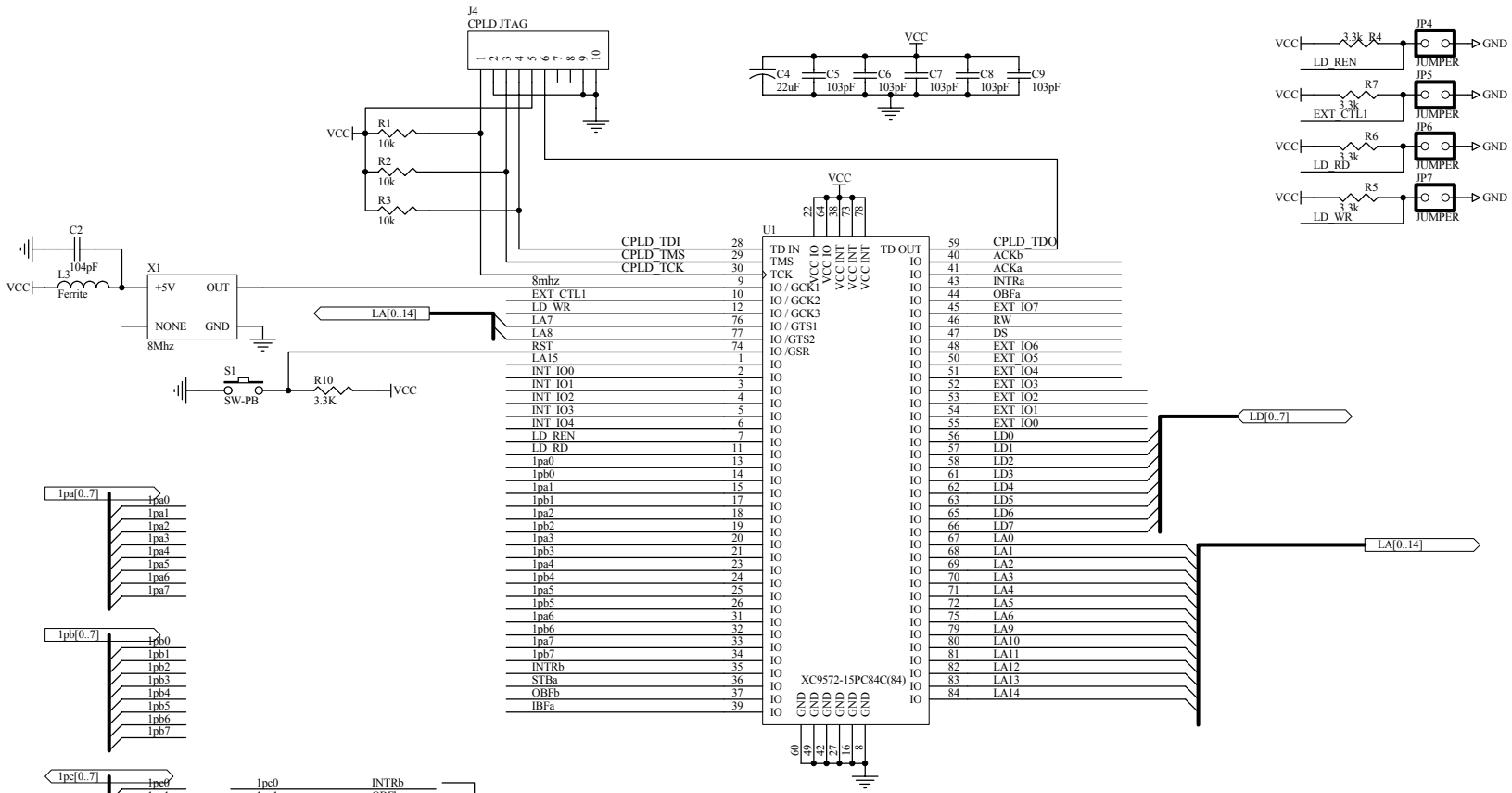




8255 I/O External Connector

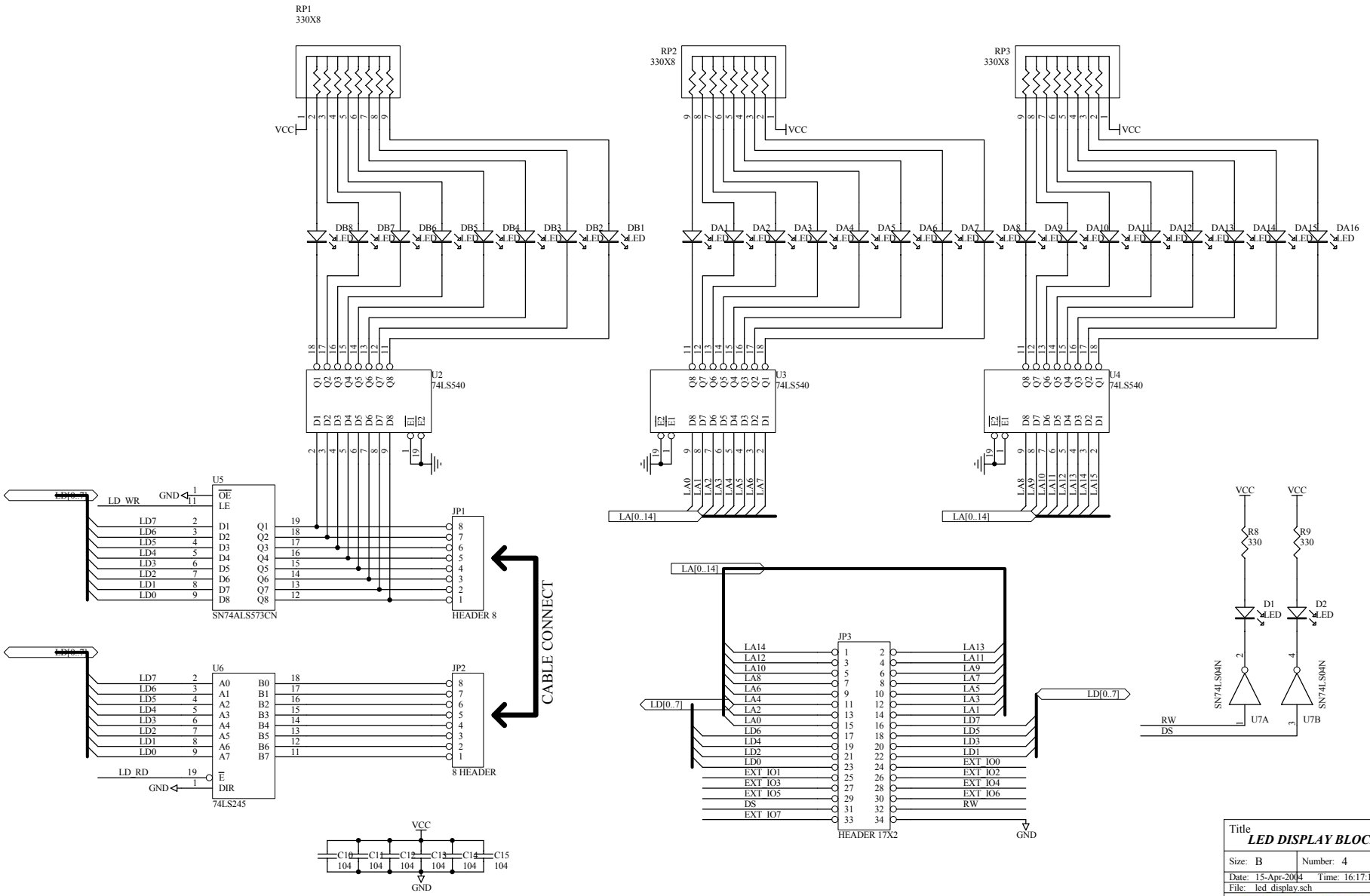
Title: ***8255 Input Connector***		Revision:
Size: B	Number:	
Date: 15-Apr-2004	Time: 16:17:12	Sheet0 of 0
File: incon.sch		

SOLITON Technologies CO., LTD.
Power by Soliton Technologies Co., LTD. Tel : 03-6566996
2F, No.198, Guangming 3rd Road, Jhubei City, Hsinchu, ROC



Title: ***Config. CPLD XC9572***	Revision: 0.0
Size: B	Number: 3
Date: 15-Apr-2004 Time: 16:17:12	Sheet3 of 4
File: xc9572.sch	

SOLITON Technologies CO., LTD.
Power by Soliton Technologies Co., LTD. Tel : 03-6566996
2F, No.198, Guangming 3rd Road, Jhubei City, Hsinchu, ROC



Title: ***LED DISPLAY BLOCK***	Revision: 0.0
Size: B	Number: 4
Date: 15-Apr-2004 Time: 16:17:12	Sheet4 of 4
File: led_display.sch	

SOLITON Technologies CO., LTD.
Power by Soliton Technologies Co., LTD. Tel : 03-6566996
2F, No.198, Guangming 3rd Road, Jhubei City, Hsinchu, ROC

```
-- 8255 Test Target : Can be used as PCI to VME A16/D8 Bridge
-- Device : 9572PC84-10
-- Functional Description :
-- 8255 Mode1, Mode2 Test
-- 8255 Configure as
-- PA : Mode 2
-- PB : Mode 1
-- Designed by : Vincent Yang   Date : 05/08/03
-- All Right Reserved by Soliton Tech.


LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY IOP_8255T IS PORT(
        -- Local I/F
        RST : in STD_LOGIC;
        OSCIN : in STD_LOGIC;
        CLK8MHz : out STD_LOGIC;

        RW : out STD_LOGIC;
        DS : out STD_LOGIC;
        LD : inout STD_LOGIC_VECTOR (7 downto 0);
        LA : out STD_LOGIC_VECTOR(15 downto 0);
        -- Local Display Control Signals
        LD_WR : out std_logic;
        LD_RD : out std_logic;
        LD_REN : in std_logic;

        -- 8255 IF
        PA : inout STD_LOGIC_VECTOR (7 downto 0);
        PB : in STD_LOGIC_VECTOR (7 downto 0);

--      INTRb : in STD_LOGIC;
        OBFb : in STD_LOGIC;
        ACKb : out STD_LOGIC;
--      INTRa : in STD_LOGIC;
        STBa : out STD_LOGIC;
--      IBFa : in STD_LOGIC;
        ACKa : out STD_LOGIC;
        OBFa : in STD_LOGIC


        );
END IOP_8255T;
ARCHITECTURE top OF IOP_8255T IS
component clkm4
  PORT(
    CLOCK: IN std_logic;
    CLK_OUT: OUT std_logic);
end component;

signal start : std_logic;
TYPE LSTATE_TYPE IS (ls1, ls2, ls3, ls4, ls5, ls6, ls7, ls8);
signal c_state: LSTATE_TYPE;
signal read_cycle : std_logic;
signal data_cycle : std_logic;
signal doe, ldoe : std_logic;
signal outdata : std_logic_vector(7 downto 0);
signal CLK, nRST : std_logic;
BEGIN

        U1 : clkm4 port map
        (
        CLOCK => OSCIN,
        CLK_OUT => CLK);
```

```
	CLK8MHz <= CLK;

	doe <= data_cycle and read_cycle;
	ldoe <= data_cycle and not read_cycle;

	PA <= LD when doe = '1' else (others => 'Z');
	LD <= outdata when ldoe = '1' else (others => 'Z');
	RW <= read_cycle;
	DS <= not data_cycle;

	LD_WR <= RST and not read_cycle and data_cycle and CLK;
	LD_RD <= not (RST and  read_cycle and data_cycle) when LD_REN = '1'
else '1';

	-- CLK Source use 8MHz
	process(RST, CLK)
	begin
		if RST = '0' then
			start <= '0';
		elsif CLK'event and CLK = '1' then
			start <= not OBFa and not OBFb;
		end if;
	end process;

	PROCESS (CLK, RST)
	BEGIN
		if (RST = '0') then
			c_state <= ls1;
			ACKa <= '1';
			ACKb <= '1';
			read_cycle <= '1';		-- Default Read Cycle
			data_cycle <= '0';
			LA <= X"0000";
			outdata <= X"00";
			STBa <= '1';
		elsif CLK'event and CLK = '1' then
			CASE c_state IS
			WHEN ls1 =>
				if start = '1' then
					c_state <= ls2;
					-- Read Address Data
					-- ACK Minimum Pulse Width = 200 ns
					ACKa <= '0';
					ACKb <= '0';
				else
					c_state <= ls1;
				end if;
				data_cycle <= '0';
			WHEN ls2 =>
				c_state <= ls3;
			WHEN ls3 =>
				c_state <= ls4;
				ACKa <= '1';
				ACKb <= '1';
				LA(14 downto 0) <= PB(6 downto 0)&PA;
				read_cycle <= PB(7);
			WHEN ls4 =>
				if (read_cycle = '1') then
					-- Read Cycle
					c_state <= ls6;
					STBa <= '0';
					data_cycle <= '1';
				else
					-- Write Cycle
					c_state <= ls5;
```

```
                        end if;
                    WHEN ls5 =>
                        -- Write Cycle, Wait for Obfa = '0'
                        if (OBFa = '0') then
                                c_state <= ls6;
                                ACKa <= '0';
                        else
                                c_state <= ls5;
                        end if;
                    WHEN ls6 =>
                        c_state <= ls7;
                    WHEN ls7 =>
                        c_state <= ls8;
                        STBa <= '1';
                        ACKa <= '1';
                        outdata <= PA;
                        if (read_cycle = '0') then
                                data_cycle <= '1';
                        end if;
                    WHEN ls8 =>
                        data_cycle <= '0';
                        c_state <= ls1;
                    WHEN others =>
                        c_state <= ls1;
                    END CASE;
            end if;
        END PROCESS;

END top;
```

```
//*******************************************************
//      File: c8255Test.cpp
//      Purpose: PCITG Mode1 and Mode2 Test Program
//
//      Author : Vincent Yang, Soliton Tech. Co. LTD.
//      Date : Mar/15/2004
//      Modification :
//      v1.00.00 : Initial
//
//*******************************************************

#include "stdafx.h"
#include "pcitg_lib.h"
#define C8255T_WR 0x0000
#define C8255T_RD 0x8000
#define C8255T_ADDRMASK 0x7FFF


int main(int argc, char* argv[])
{
        int board_index = 0;
        pcitg_t board0;
        unsigned long data;
        unsigned long address, aH, aL;
        int execnt = 0;
        int sleeptime = 10;


        if (argc > 1)
        {
                board_index = atoi(argv[1]);
        }
        if (argc > 2)
        {
                sleeptime = atoi(argv[2]);
                if (sleeptime < 1) sleeptime = 1;
        }

        board0 = pcitg_open(board_index);
        if (board0->hDevice == PCITG_NULL)
        {
                printf("ERROR opening Board[%d]: (%0x) returned from
CreateFile\n", board_index, GetLastError());
                return 1;
        }
        else
        {
                printf("Board[%d] found, Device Handle = 0x%.8X\n",
board_index, board0->hDevice);
        }

        // Need to initialize hardware
        // Hardware might stay in some unknown state
        pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_ALL, 0xFFFFFF80);
        pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_C, 0x7D);
        pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_C, 0xFF);


        // PA : Mode 2
        // PB : Mode 1 Output
        // Ctl = 0x11XXX10X = 0xC4
        pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_CTL, 0xC4);

        address = 0x0000 | C8255T_WR;
        data = 0x55;
```

```
        aH = (address & 0xFF00) >> 8;
        aL = address & 0xFF;
        pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_B, aH);
        pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_A, aL);
        pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_A, data);

        printf("%s A[0x%.4X] D[0x%.2X]\n", (address &
C8255T_RD)?"Read":"Write", (address & 0x7FFF), (data & 0xFF));

//      printf("Press any key to continue.....\n");
//      getch();


        address = 0x5555 | C8255T_RD;

        aH = (address & 0xFF00) >> 8;
        aL = address & 0xFF;
        pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_B, aH);
        pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_A, aL);
        pcitg_in(board0, PCITG_CHIP0, PCITG_PORT_A, &data);

        printf("%s A[0x%.4X] D[0x%.2X]\n", (address &
C8255T_RD)?"Read":"Write", (address & 0x7FFF), (data & 0xFF));

//      printf("Press any key to continue.....\n");
//      getch();

        address = 0x1234 | C8255T_WR;
        data = 0xAA;

        aH = (address & 0xFF00) >> 8;
        aL = address & 0xFF;
        pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_B, aH);
        pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_A, aL);
        pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_A, data);
        printf("%s A[0x%.4X] D[0x%.2X]\n", (address &
C8255T_RD)?"Read":"Write", (address & 0x7FFF), (data & 0xFF));

//      printf("Press any key to continue.....\n");
//      getch();

        address = 0x2AAA | C8255T_RD;

        aH = (address & 0xFF00) >> 8;
        aL = address & 0xFF;
        pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_B, aH);
        pcitg_out(board0, PCITG_CHIP0, PCITG_PORT_A, aL);
        pcitg_in(board0, PCITG_CHIP0, PCITG_PORT_A, &data);

        printf("%s A[0x%.4X] D[0x%.2X]\n", (address &
C8255T_RD)?"Read":"Write", (address & 0x7FFF), (data & 0xFF));

//      printf("Press any key to continue.....\n");
//      getch();


        pcitg_close(board0);

        return 0;
}
```