ICS ELECTRONICS
*division of Systems West Inc.*

*Better Solutions for GPIB and VXIbus Systems*

VXIbus APPLICATION BULLETIN

# Interfacing a VXI-5526 VXI Interface Card to an ISA Card

## INTRODUCTION

ICS's VXI-5526 Interface Card has been designed so it can interface virtually any user's circuit to the VXIbus. The problem often encountered by designers when adapting all or part of an existing ISA bus card schematic from a PC to the VXIbus is how to make a quick connection to test the concept and work out the programming problems prior to doing the actual schematic and PC board. This application note shoes how an ISA bus card can be quickly interfaced to the VXIbus using a VXI-5526 Interface Card. The approach described in this application note can be used to interface virtually any type of a bus based card to the VXIbus using a VXI-5526 Interface Card.

## PROBLEM DEFINITION

The problem in connecting an ISA bus compatible card to the VXIbus with a VXI-5526 is the physical connections and then the selection of the commands to control the card. This application note deals with the hardware interconnect problem and describes some guidelines for controlling the card.

The solutions shown in this application note were derived from our experiences in connecting a 488-PC2 GPIB Controller Card to the VXI-5526 to debug DMA command functions before laying out the PCB for the Model VXI-5538 Dual GPIB Bus Controller Module.

## HARDWARE INTERFACE

A block diagram of the VXI-5526 VXI Interface Card is shown in Figure 1. The VXI-5526 provides the user with 32 static digital signals, a 386EX expansion bus and a serial interface. The 386EX expansion bus includes a 16 bit wide data bus, a 6 bit address bus, IOR and IOW signals plus an address strobe, DMA control signals and two IRQ (interrupt) inputs. The 386EX's expansion bus signals are very similar to those found on the PC ISA bus. The Chip Select signal, SCS3#, is asserted whenever the expansion bus is active and is similar to the AEN/ signal on the ISA bus. VXI-5526 signals with The expansion bus can directly access 64 I/O addresses between 300 Hex and 33F Hex. All of the VXI-5526's User Interface signals are listed in Table 1.

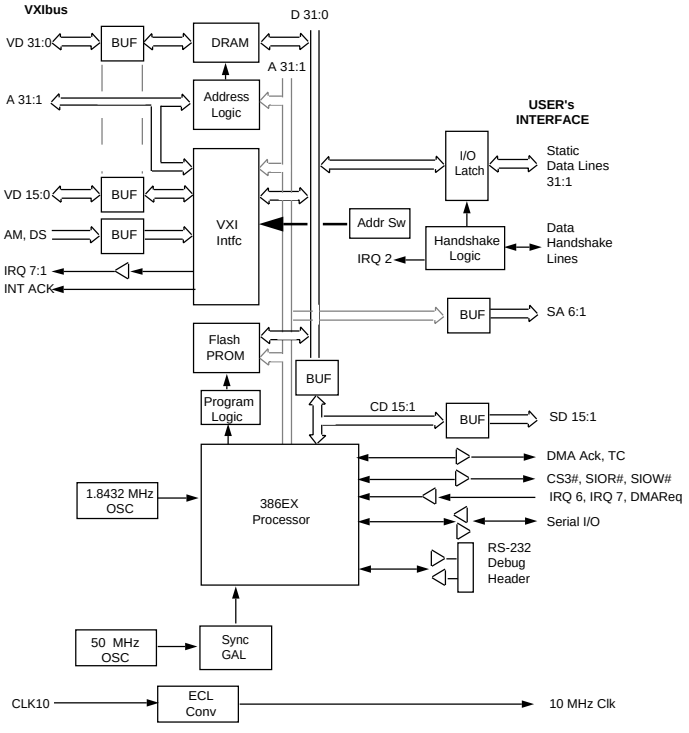


**Figure 1 VXI-5526 Block Diagram**



*(over)*

## TABLE 1 VXI-5526 USER INTERFACE SIGNALS

| Signal | Pin No. | Function | Signal | Pin No. | Function |
|---|---|---|---|---|---|
| CH32 | A22 | bit D15 Most Significant bit | SD11 | A8 | |
| CH31 | A23 | bit D14 | SD10 | A9 | |
| CH30 | A24 | bit D13 | SD9 | A10 | |
| CH29 | A25 | bit D12 | SD8 | A11 | |
| CH28 | A26 | bit D11 | SD7 | A12 | |
| CH27 | A27 | bit D10 | SD6 | A13 | |
| CH26 | A28 | bit D9 | SD5 | A14 | |
| CH25 | A29 | bit D8 | SD4 | A15 | |
| CH24 | B21 | bit D7 | SD3 | A16 | |
| CH23 | B23 | bit D6 | SD2 | A17 | |
| CH22 | B24 | bit D5 | SD1 | A18 | |
| CH21 | B25 | bit D4 | SD0 | A19 | Expansion Data Bus - LSB |
| CH20 | B26 | bit D3 | | | |
| CH19 | B27 | bit D2 | SA6 | B4 | Expansion Bus Adress line |
| CH18 | B28 | bit D1 | SA5 | B5 | |
| CH17 | B29 | bit D0 Least Significant bit | SA4 | B6 | |
| CH16 | C14 | bit D15 Most Significant bit | SA3 | B7 | |
| CH15 | C15 | bit D14 | SA2 | B8 | |
| CH14 | C16 | bit D13 | SA1 | B9 | Expansion Bus Adress line - lsb |
| CH13 | C17 | bit D12 | | | |
| CH12 | C18 | bit D11 | CS3# | B13 | Expansion Bus Access strobe |
| CH11 | C19 | bit D10 | IOR# | B11 | Read Strobe output |
| CH10 | C20 | bit D9 | IOW# | B10 | Write Strobe output |
| CH9 | C21 | bit D8 | - | B17 | spare |
| CH8 | C22 | bit D7 | UASEL# | B16 | Upper VXI Address Select |
| CH7 | C23 | bit D6 | IOCHRDY# | C7 | |
| Ch6 | C24 | bit D5 | IOSC16# | A20 | |
| CH5 | C25 | bit D4 | | | |
| CH4 | C26 | bit D3 | DMAREQ | B19 | DMA Request input |
| CH3 | C27 | bit D2 | DMAACK# | B18 | DMA Acknowledge output |
| CH2 | C28 | bit D1 | TC | B20 | Terminal Count output |
| CH1 | C29 | bit D0 Least Significant bit | | | |
| | | | IRQ6 | B14 | Interrupt input |
| EDR | C5 | External Data Ready input | IRQ7 | B15 | Interrupt input |
| Inhibit# | C4 | Data Inhibit output | SOUT | C13 | Serial Data output |
| Strobe# | C3 | Output Data Strobe output | SIN | C12 | Serial Data input |
| Clear# | A21 | Clear pulse output | SRTS# | C11 | Request-to-send output |
| TTLTrigger# | C8 | Selected VXIbus TTL Trigger line | SCTS# | C10 | Clear-to-send input |
| CLK16 | C32 | 16 MHz Clock output | SDSR# | C9 | Data-set-ready input |
| CLK10 | C6 | VXIbus 10 MHz clock signal | | | |
| | | | VCC | B1,C1 | +5 Vdc |
| RDY LED# | C31 | LED driver output | -5.2V | B3 | |
| ACCESS LED# | C30 | LED driver output | -2V | A3 | |
| FAILED LED# | B30 | LED driver output | +12V | A2 | |
| SYSFAIL LED# | A30 | LED driver output | -12V | A1 | |
| RESET SW# | B32 | Reset switch input | +24V | A32 | |
| | | | -24V | A31 | |
| SD15 | A4 | Expansion Data Bus - MSB | GND | B2,B12, B22,B31, C2 | Signal ground |
| SD14 | A5 | | | | |
| SD13 | A6 | | | | |
| SD12 | A7 | | | | |

Notes: Signals ending with an # are low true.

The card to be interfaced is a Model 488-PC2 Card which is a 8-bit ISA bus card. It uses three I/O address lines, can transfer data with DMA and can generate an interrupt on a selected IRQ line. The 488-PC2's base address is 2E1 and it has eight I/O registers at increments of 400 Hex.

Any bus card can be directly connected to the VXI-5526 if it uses 16 or less data lines and 64 or less register or I/O port addresses. It doesn't matter whether the registers were originally I/O mapped or memory mapped. When connected to the VXI-5526, they will all be I/O mapped. All data lines are directly connected to the VXI-5526 data lines so that the least significant bits match. Any unused VXI-5526 data lines can be left open. All bus card address and control lines should be connected to a signal or voltage source. The card's static address and control signal lines can be tied directly to Ground or to Vcc through a pullup resistor to establish the card's normal base address. The active address lines should be connected to the VXI-5526's address lines so that the lowest active address line is connected to the VXI-5526's SA1 address line. Unused VXI-5526 address lines are left open.

In the case of ICS's 488-PC2 Card, the 488-PC2 Card only uses the lower eight data lines and 8 register addresses.

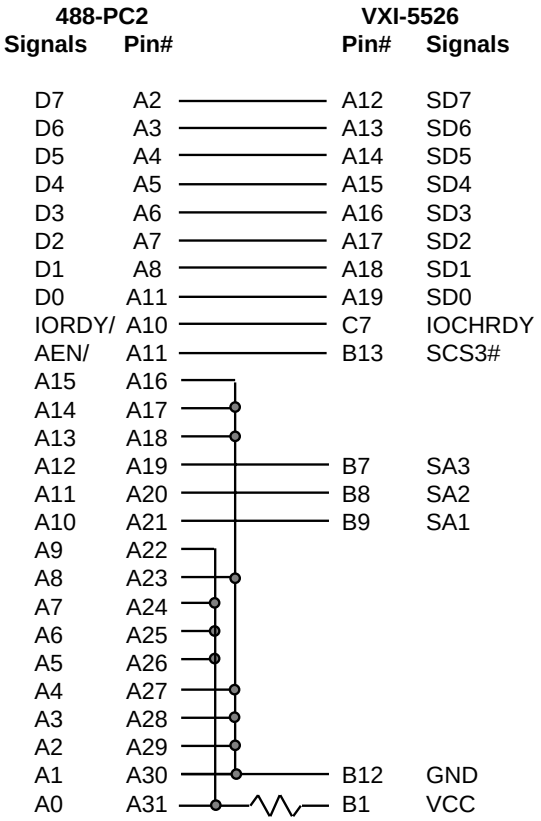


**Figure 2 VXI-5526 to 488-PC2 Connections**

Therefore, only the lower eight VXI-5526 data lines are connected to the 488-PC2's data lines. The remaining VXI-5526 data lines are left open. To allow the VXI-5526 to address the 488-PC2 registers, the three lower VXI-5526 address lines are connected to the 488-PC2 register address lines as shown in Figure 2. Because the 488-PC2's active address inputs do not line up with the VXI-5526 address lines, the difference in the address line values will be taken care of during the programming phase by placing defines in the header file. The 488-PC2's static address lines are jumpered to Vcc or to ground to create an address of 2E1 Hex. The VXI-5526 chip select signal, SCS3#, is connected to the 488-PC2's AEN/ input. On the ISA bus, the AEN/ signal is asserted when the address bus is valid and is not being used for DMA data transfers. The SCS3# signal on the VXI-5526 expansion bus serves the same purpose. Figure 2 shows the completed VXI-5526 to 488-PC2 Card connections.

If DMA transfers were being implemented with Fast Data Channels, the user should only assert DMA when UASEL# is not asserted to prevent bus conflicts. Once DMA is asserted, it should be held on until DMAACK# is received.

## TEST PROGRAM

There are several ways to program the VXI-5526 to operate circuits or boards connected to the User's Interface. One method is to use the VXI-5526's general command set to control the User's Interface. This method uses word serial messages sent to the VXI-5526 from the Slot 0 Controller. An example for this method will be shown when the commands are finalized.

A second method is to write a special program for the VXI-5526 that is run by the 386EX processor on the VXI-5526 board. This method is applicable to users with the VXI-5526 Firmware Development Kit. This second method was used to test the DMA control lines with the 488-PC2 card.

## PROGRAM DESCRIPTION

The DMA test program consisted of a main program, a header file, and other routines to initialize the 488-PC2 and the 386EX processor. The main routine called the initialization functions and then read switches on the 488-PC2 card to branch to a test loop. The different test loops were selected by setting the switches on the 488-PC2 card. These routines performed DMA data transfer tests to validate DMA functions that were being written for the VXI-5526 card. Figure 3 shows the main routine.

The header file, pc2_io.h, defines the constants used by the other routines. In this case, the header file is similar to the one used in ICS's PC2 C language library. In a standard 488-PC2, the register addresses are in 400 Hex increments starting at 0x2E1. i.e. 0x2E1, 0x6E1, 0AE1, etc. When the 488-PC2 card is connected to the VXI-5526, the register addresses were redefined to be in 0x2 Hex increments starting at 0x300. i.e. 0x300, 0x302, 0x304 etc. The relevant portion of the header file is listed in Figure 4.

The VXI-5526 is designed for 16 bit data transfers and each read or write on the expansion bus are performed as a 16 bit reads or writes. The user should take care when inputting 8 bit data to mask off the unused upper eight bits before doing any calculations on the input value. Eight bit output data should always be justified to the least significant bit.

Contact ICS Electronics for a copy of the DMA test program.

```
/* ********************************************************** */
/*                    5526 DMA Test                           */
/*    filename: dmatest.c                                     */
/*    5526_CT DMA 0 initialization functions for DMA testing  */
/*                                                            */
/*    Created:   03/11/96 BG -                                */
/*    Revised:                                                */
/*                                                            */
/*    Program sets up the 386EX, the 7210, and DMA channel
      0, then                                                 */
/*              puts itself into an infinite loop.            */
/* ********************************************************** */
```

**Figure 3 5526 DMA Test Main Routine**

```
/* Includes */
#include <conio.h>
#include "80386ex.h"

/* Function declarations */
void Init_DMA0(unsigned char lo_byte, unsigned char hi_byte,
unsigned char mode);
void InitMemToMemDMA(void);
void Send_MemToMem(void);
void Init_CSU(void);
void init_7210(void);
short test_7210(void);
void Init_DMA_7210(void);
unsigned char *data, *comp;

void main(void)
{
      // Local variables
      short option,i;
      unsigned char value;

      Init_CSU(); // Set up the intel 386 chip select unit

      outpw (0x222,0x06);    // If no error, Light Ready LED

      value = 0x31;
      i = 0;
      option = inpw(0x8620);     //Read switches on board
      if (option & 0x02)              //If switch 2, do mem to mem xfr
            {
            InitMemToMemDMA();
            data = (unsigned char *)0x70000L;
            for (i=0;i<=0xFF;i++)
                  *(data++)=i;
            for (i=0;i<=0xFF;i++)
                  Send_MemToMem();
            data = (unsigned char *)0x70000L;
            comp = (unsigned char *)0x80000L;
            i=0;
            while(data++==comp++)
                  i++;
            if(i==256)
                  outp(0x222,0x05);
            else
                  outp(0x222,0x03);
            }
      else
      if (option & 0x01)            //If switch 1, do continuous DMA
            {
            Init_DMA0(0xFF,0xFF,0x54); // Initialize DMA
            channel 0 to do 64K byte at 0x80000
            init_7210(); // Initialize 7210
            Init_DMA_7210();
            while (1)       //infinite loop
                  i++;
      }
      else                        //other, xfr block and verify block xfr
      {
//          while(1)
//          {
                  Init_DMA0(0x05,0x00,0x44);
                  init_7210(); // Initialize 7210
                  Init_DMA_7210();
                  while (!(inpw(DMASTS) & 0x01) )
                  ;//Wait for Transfer complete
                  data = (unsigned char *)0x80000L;
                  while ( *(data++) == value++)
                        i++;
                  if ((*(—data)==0x0A) && (i == 10))
                        outp(0x222,5);
                  else
                        outp(0x222,3);
                  for(i=0;i<20000;i++)
                        i=i;
//          }
      }
}
```

**Figure 3 5526 DMA Test Main Routine (Cont'd)**

**Figure 3 5526 DMA Test Main Routine (Cont'd)**

```
/* ********************************************************************* */
/*                  PC2 I/O ADDRESSES                                    */
/*                                                                       */
/*  Filename:  pc2_io.h                                                  */
/*                                                                       */
/*  This file contains the I/O addresses for the PC2 card or             */
/*  the VXI-554x VXI controllers                                         */
/*  Modified for use with 5526 DMA Testing, with DMA test
      cable                                                            */
/*                                                                       */
/*  Created:  02/18/94 CB                                                */
/*  Modified:  03/19/96                                                  */
/* ********************************************************************* */

//     GPIB registers (NEC 7210)
//          Read Registers
#define Data_In          0x0300     // 0R - Data In
#define Int_Stat1        0x0302     // 1R - Interrupt Status 1
#define Int_Stat2        0x0304     // 2R - Interrupt Status 2
#define SPoll_Stat       0x0306     // 3R - Serial Poll Status
#define Addr_Stat        0x0308     // 4R - Address Status
#define Cmd_Pass         0x030A     // 5R - Command Pass
Through
#define Address_0        0x030C     // 6R - Address 0
#define Address_1        0x030E     // 7R - Address 1

//          Write Registers

#define Data_Out         0x0300     // 0W - Data Out
#define Int_Mask1        0x0302     // 1W - Interrupt Mask 1
#define Int_Mask2        0x0304     // 2W - Interrupt Mask 2
#define SPoll_Mode       0x0306     // 3W - Serial Poll Mode
#define Addr_Mode        0x0308     // 4W - Address Mode
#define Aux_Mode         0x030A     // 5W - Auxiliary Mode
#define Address01        0x030C     // 6W - Address 0 / 1
#define EOS_reg          0x030E     // 7W - End of String

#define Cmd_Status       0x82e1     // Read Only - Command
Line Status
```

**Figure 4 Portion of pc2-io.h Header File (showing register defines)**