

Application Note 22

Wing Poon

August 1999

The Industry Standard Architecture (ISA) bus is the bus of pre-eminence (until very recently) in IBM-PC compatible computers. It has since been adopted as the IEEE P996 standard, and appears in other derivative standards such as PC/104 and PCMCIA.

The initial ISA bus was really an extension of the first IBM-PC's microprocessor bus: an Intel 8088 running at 4.77MHz. When the IBM-AT microcomputer emerged, the bus was modified for use with the Intel 80286 microprocessor running at 8MHz.

The 80286 had 16 external data signals, compared to the 8088's 8. Also, the 80286 introduced pipelined addressing. These two facets make interfacing the ISA-AT (16-bit) quite a bit more complex than interfacing to the ISA-XT (8-bit) bus. This document addresses the ISA-XT bus.

Since the ISA bus is an extension of the microprocessor bus, we would find essentially three groups of signals on this bus: address, data and control.

The ISA-XT has 20 address lines: SA0-SA19. Thus it is able to address 1Mbyte of data memory. The 8088 had separate address spaces for memory versus I/O. The I/O address space is defined only by SA0-SA15. Thus the 8088 can address 64kBytes of I/O registers. However, the designers of the IBM-PC thought that was way too generous and passed a resolution that only 0100h-03FFh were valid I/O addresses in the PC architecture, hence reducing the cost of peripherals since then, designers only need to look at A0-A9.

The data bus consists of SD0-SD7. This is a bi-directional bus.

The control signals can be further subdivided into the following subgroups: read/write control, DMA control and interrupts.

The ISA-XT bus connector has 62 pins; fortunately, for a simple application, we only need to look at 22 signals: SA0-9, SD0-7, IOWC*, IORC*, CHRDY and RESET.

Refer to Appendix A for the schematics of our demonstration circuit.

In this example, we will implement 4 readable and writable registers mapped to a user-selectable I/O address range (we recommend 0300h-0303h). The PC can then read and write to these registers to effect functions on the

SX according to some pre-defined protocol. The circuit can be modified very easily to implement more than 4 registers, if necessary.

There are two approaches to interfacing the SX to the ISA bus: real-time/no-wait-state or handshake/wait-stated. This document describes the handshake/wait-stated implementation.

The SX48/52BD are among the very few 8-bit micro-controllers fast enough to implement a real-time/no-wait-state interface method. While this method will allow the SX48/52BD (the SX18/28AC do not have enough I/O pins for this method) to directly connect to the ISA bus, it imposes severe limitations on the software designer's flexibility in writing software on the SX, primarily because the SX needs to respond to bus-activity in a real-time fashion.

The handshake/wait-stated approach does require the use of external glue-logic, but a very minimal amount – in our case, just two 74xx type ICs (~\$0.50 total). U1 is an 8-bit equality comparator and U2 is a quad two-input open-collector NAND gate.

U1 is used to decode the most-significant eight bits of the I/O address: SA[9:2]. The (P=Q)* (which we'll also call CS*, for chip-select) will be asserted if and only if two conditions are met:

- The most-significant eight address bits, SA[9:2] match some pre-defined pattern
- An I/O read or write cycle is in progress

The second condition is necessary because the same set of address lines is used for both memory accesses (which we do not want to participate in) and I/O accesses (which we do want to participate in).

U3a implements a negative-logic OR function on the read and write strobes, IORC* and IOWC* respectively, and allows CS* to become asserted only if either of the former signals are asserted.

U1's Q[0:6] select the seven most-significant bits, [9:3], of the desired address range. The design imposes a restriction that bit[2] of the desired address range must be zero. Bits[1:0] are decoded by the SX to refer to each of the four addressable registers on the chip.

Scenix™ and the Scenix logo are trademarks of Scenix Semiconductor, Inc.
All other trademarks mentioned in this document are property of their respective companies.

U3b and U3d are used to generate the handshaking signal, CHRDY. CHRDY is used to interface slower devices onto the ISA bus. When deasserted, the PC will add wait-states to the current bus cycle (read or write, I/O or memory), with no upper-bound limitation. CHRDY is a wired-OR signal and is pulled high via a pull-up resistor on the motherboard.

This section of the circuit can be taken out if the only source of interrupt on the SX is a single Port B pin edge interrupt. The SX, with its deterministic and fast interrupt response, will be able to transact a bus cycle with zero wait-states, if necessary. However, since many of our customers use other sources of interrupts (e.g. RTCC), we have added the handshaking circuitry to obviate the need for real-time response to bus activity, thus making the SX firmware designer's job much easier.

Pin12 of U3d is what we'll call the CHRDY_CTRL signal. This output from the SX determines whether the handshake circuitry is primed (CHRDY_CTRL = high), or whether the SX is temporarily overriding the CHRDY signal (CHRDY_CTRL = low).

Here is how the handshaking is done:

Normally, Pin12 of U3d (CHRDY_CTRL) is high, so when CS* goes low, indicating a read/write access to the SX, CHRDY is immediately deasserted, thus stalling the PC's processor. When the SX gets around to checking the CS* line (via RB0), it looks at IOWC* and IORC* to determine what kind of bus cycle is in progress, looks at SA[1:0] to determine which register to access, and either samples SD[0:7] (if a write cycle is indicated) or outputs data on SD[0:7] (if a read cycle is indicated). When this has been done, it will set CHRDY_CTRL low which will force CHRDY to be re-asserted. Then, the SX will continuously monitor CS* to determine when it can exit from this override condition. When CS* is deasserted, the SX sets CHRDY_CTRL back high, which primes the handshaking circuitry for the next I/O cycle.

PortC (SD[0:7]) is normally configured as an input, so effectively disconnecting itself from the ISA bus, thus preventing bus contention. It is set as an output only during a read cycle, and then again, only temporarily.

It is worth noting that the chances of a glitch on CS* is minimal. This is because the comparator's Q input is always constant while its P input consists of two groups of signals, {P0-6,G} and {P7}, that change at non-overlapping times. This is further mitigated by the SX firmware sampling CS* twice before a decision is made.

With the handshaking circuitry, the software designer can choose to use either an edge-triggered interrupt or polling mechanism (off the RTCC interrupt, for example) to complete the bus transaction. In the example code given in Appendix B, RB0 is configured as a negative-edge triggered interrupt. However, you are free to implement an approach whereby you periodically poll RB0 (CS*) and call the ISA code whenever RB0 reads '0'.

You now have a very versatile interface that can be used to control and communicate with other Virtual Peripherals incorporated in the SX communications controller. Refer to the document "An ISA to Single-channel I2C Multi-Master Implementation" for an example of what can be done.

1.0 Appendix A

```

; ISA_SX.asm
; Wing Poon . 7/28/99 . (C) Scenix Semiconductor, Inc.
; This code demonstrates how to interface an SX to the ISA-XT Bus.
; It implements 4 readable and writeable registers that are mapped to
; low-order address bits A1:A0.
; This implementation uses the PortB edge-triggered interrupt mechanism
; to trigger on a falling edge of CS*, but a polling mechanism can
; also be used.

; *****
; *** DEVICE ***
; *****

                DEVICE    PINS28,PAGES4,BANKS8,OSCHS,TURBO,STACKX,OPTIONX
                RESET     main

; *****
; *** VARIABLES ***
; *****

; *** Global ***
                ORG        $08
REG0            DS        1
REG1            DS        1
REG2            DS        1
REG3            DS        1
ISR_MODE_SAVE   DS        1

; *****
; *** EQUATES ***
; *****

RA_DIR          =          %0000
RB_DIR          =          %00111101
RC_DIR_IN       =          %11111111
RC_DIR_OUT      =          %00000000

WKED_B          =          %00000001
WKEN_B          =          %11111110

DEBUG_CLK1      =          ra.0           ; debugging use
DEBUG_DATA1     =          ra.1           ; debugging use

CS              =          rb.0           ; chip-select (decode SA9:SA2)
CHRDY_CTRL      =          rb.1           ; CHRDY (channel-ready) arm/override
SA0             =          rb.2           ; ISA: System Addr[0]
SA1             =          rb.3           ; ISA: System Addr[1]
IORC            =          rb.4           ; ISA: I/O Read Control (active-low)
IOWC            =          rb.5           ; ISA: I/O Write Control (active-low)

SD              =          rc             ; ISA: System Data[0:7]

; *****
; *** MACROS ***
; *****

; *****
; *** ISR ***
; *****

                ORG        0

```

```

isr
    ; save MODE register (cus we'll be changing it)
    mov     w, m
    mov     ISR_MODE_SAVE, w

    mode    #$9                ; WKPND_B
    mov!    rb, #0             ; clear int pending bit

    ; point FSR to the addressed register
    mov     fsr, #REG0
    snb     SA0
    inc     fsr
    mov     w, #2
    snb     SA1
    add     fsr, w

    sb      IOWC
    jmp     :writeCycle        ; IOWC* is active-low
    sb      IORC
    jmp     :readCycle         ; IORC* is active-low
    jmp:isrExit

:writeCycle
    mov     indf, SD           ; store data
    clrb    CHRDY_CTRL        ; force CHRDY high
:writeWait
    sb      CS
    jmp     :writeWait         ; wait for PC to acknowledge cycle completion
    setb    CHRDY_CTRL        ; prime CHRDY for action again
    jmp     :isrExit

:readCycle
    mode    #$F                ; Port Direction
    mov     !rc, #RC_DIR_OUT   ; configure as output
    mov     SD, indf           ; put out data
    clrb    CHRDY_CTRL        ; force CHRDY high
:readWait
    sb      CS
    jmp     :readWait          ; wait for PC to acknowledge cycle completion
    setb    CHRDY_CTRL        ; prime CHRDY for action again
    mov     !rc, #RC_DIR_IN    ; configure back as input
    jmp     :isrExit

:isrExit
    ; restore MODE register
    mov     w, ISR_MODE_SAVE
    mov     m, w

    reti

; *****
; *** MAIN PROGRAM ***
; *****

main
    ; configure the device
    mode    #$F                ; Port Directions
    mov     !ra, #RA_DIR
    mov     !rb, #RB_DIR
    mov     !rc, #RC_DIR_IN
    clr     ra
    clr     rb
    clr     rc
    mode    #$A                ; WKED_B
    mov     !rb, #WKED_B

```

```

        mode    $$9                ; WKPND_B
        mov     !rb, #0
        mode    $$B                ; WKEN_B
        mov     !rb, #WKEN_B

        mov     !option, #01111111 ;enable W addressing

        clr     REG0
        clr     REG1
        clr     REG2
        clr     REG3

        setb    CHRDY_CTRL         ; prime CHRDY for action

mainLoop
; Once initialized, the main code really doesn't need to do anything
; since the ISA interface code is entirely interrupt-driven.
; Here we output the contents of each register in turn in a synchronous
; bit-bang fashion just so as to verify the contents of the registers
        mov     w, REG0
        call    outReg
        mov     w, REG1
        call    outReg
        mov     w, REG2
        call    outReg
        mov     w, REG3
        call    outReg

        mov     w, #255
        call    delay
        jmp     mainLoop

; *****
; *** SUBROUTINES ***
; *****

outReg
; INPUT: W = byte to be shifted out (MSb first)
; OUTPUT: none

        REPT    8
        movb    DEBUG_DATA1, wreg.7
        setb    DEBUG_CLK1
        rl      wreg
        nop
        clrb    DEBUG_CLK1
        ENDR
        ret

delay
; INPUT: W = delay count
; OUTPUT: none

        decsz   wreg
        jmp     delay
        ret

; *****
; *** LOOKUP TABLES ***
; *****

; *****
; *** END ***
; *****

        END

```

Lit#: SXL-AN22-01

Sales and Tech Support Contact Information

For the latest contact and support information on SX devices, please visit the Scenix Semiconductor website at www.scenix.com. The site contains technical literature, local sales contacts, tech support and many other features.

SCENIX

**1330 Charleston Road
Mountain View, CA 94043**

Tel.: (650) 210-1500

Fax: (650) 210-8715

E-Mail: sales@scenix.com

Web Site: www.scenix.com