INTRODUCTION

The PIC16C64/C74 microcontroller from Microchip Technology Inc., can be interfaced in a multi-microprocessor environment with ease using the built-in Parallel Slave Port. With their very high operating speeds (cycle times as low as 200ns with a clock rate of 20MHz), and an array of on-chip peripherals, they make ideal smart interfaces to the real world.

IMPLEMENTATION

PortD operates as an 8-bit wide parallel slave port, with PortE providing the control signals when bit PSPMODE (TRISE<4>) is set. In parallel slave mode, PortD is asynchronously readable and writable by the external world through the CS (RE2/CS), RD (RE0/RD), and WR (RE1/WR) control inputs.

In order to use the parallel slave port, the data direction bits in the TRISE register corresponding to RD, WR, and CE (TRISE<2:0>) must be configured as inputs (set = 1). The port pins are connected to two 8-bit latches, one for data output (from the PIC16CXX) and one for data input. The PIC16CXX sends data by writing to the output latch, and receives data by reading the input latch (note that the input and output latches are at the same address). In this mode the TRISD register is ignored, since the external device connected to the slave port controls the direction of data flow.

When the external device performs either a read or a write operation to the PIC16CXX, the interrupt flag, PSPIF (PIR1<7>), will be set and the processor interrupted if PSPIE (PIE1<7>) is set and interrupts are enabled (GIE and PEIE, (INTCON<7:6>) set). When the interrupt is serviced, PSPIF must be cleared by software.

The read-only status flag IBF, Input Buffer Full (TRISE<7>), is set if a received word is waiting to be read. IBF is cleared upon read of the input buffer latch. If another word is received prior to the first being read, status flag IBOV (TRISE<5>) is set. IBOV can be cleared by software.

The Output Buffer Full status bit, OBF (TRISE<6>), is set if a word written to PortD latch is waiting to be read by the external bus.

When not in PSPMODE the IBF and OBF bits are cleared. If the IBOV flag was previously set, however, it must be cleared by software.

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Function</th>
<th>Address</th>
<th>Power-On Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORTD</td>
<td>Parallel slave port Read/Write Data</td>
<td>08h</td>
<td>XXXXX XXXX</td>
</tr>
<tr>
<td>TRISD</td>
<td>PortD data direction register</td>
<td>88h</td>
<td>111 11 11</td>
</tr>
<tr>
<td>PORTE</td>
<td>Parallel slave port Read/Write/Chip Select signals</td>
<td>09h</td>
<td>- - - - X X X</td>
</tr>
<tr>
<td>TRISE</td>
<td>Control bits for PortD slave port</td>
<td>89h</td>
<td>0000 - 111</td>
</tr>
<tr>
<td>INTCON</td>
<td>Global Interrupt Enable</td>
<td>0Bh</td>
<td>000 0 0 0 X</td>
</tr>
<tr>
<td>PIR1</td>
<td>Interrupt register (PSPIF bit)</td>
<td>0Ch</td>
<td>000 0 0 0</td>
</tr>
<tr>
<td>PIE1</td>
<td>Interrupt Enable register (PSPIE bit)</td>
<td>8Ch</td>
<td>000 0 0 0</td>
</tr>
</tbody>
</table>
Using the 8-Bit Parallel Slave Port

FIGURE 2: PORT E FUNCTIONS

<table>
<thead>
<tr>
<th></th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>U</td>
<td>U</td>
<td>U</td>
</tr>
</tbody>
</table>

- RE0/RD/AN5 - Input/output port pin. Read control input in parallel slave port mode, or analog input.
  - RD
  - 1 = Not a read operation
  - 0 = Read operation. The system reads the PIC16CXX PortD register (if chip selected)

- RE1/WR/AN6 - Input/output port pin. Write control input in parallel slave port mode, or analog input.
  - WR
  - 1 = Not a write operation
  - 0 = Write operation. The system writes to the PIC16CXX PortD register (if chip selected)

- RE2/CS/AN7 - Input/output port pin. Chip select control input in parallel slave port mode, or analog input
  - CS
  - 1 = Device is not selected
  - 0 = Device is selected
# FIGURE 3: TRISE REGISTER

<table>
<thead>
<tr>
<th>R</th>
<th>R</th>
<th>R/W</th>
<th>R/W</th>
<th>U</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBF</td>
<td>OBF</td>
<td>IBOV</td>
<td>PSPMODE</td>
<td>-</td>
<td>TRISE2</td>
<td>TRISE1</td>
<td>TRISE0</td>
<td></td>
</tr>
</tbody>
</table>

- **TRISE0**: Direction control bit for port pin RE0
  - 1 = Input
  - 0 = Output
- **TRISE1**: Direction control bit for port pin RE1
  - 1 = Input
  - 0 = Output
- **TRISE2**: Direction control bit for port pin RE2
  - 1 = Input
  - 0 = Output
- **Unimplemented**: read as ‘0’
- **PSPMODE**: Selects parallel slave port mode for PortD and PortE
  - 1 = Parallel slave port mode
  - 0 = General purpose I/O.
- **IBOV**: Input buffer overflow in microprocessor mode.
  - 1 = A write occurred before the previous input word was read.
  - 0 = No overflow has occurred.
- **OBF**: Output buffer full.
  - 1 = The output buffer still holds a previously written word.
  - 0 = Output buffer has been read.
- **IBF**: Input buffer full.
  - 1 = A word has been received and is waiting to be read by the CPU.
  - 0 = No word has been received.
Using the 8-Bit Parallel Slave Port

**FIGURE 4: PIE1 REGISTER**

<table>
<thead>
<tr>
<th></th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSPIE</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSPIE</td>
<td>CCP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
</tr>
</tbody>
</table>

**PSPIE**: Parallel slave port interrupt enable bit.
- 1 = Enables PSPIF interrupt.
- 0 = Disables PSPIF interrupt

**FIGURE 5: PIR1 REGISTER**

<table>
<thead>
<tr>
<th></th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSPIF</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSPIF</td>
<td>CCP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**PSPIF**: Parallel slave port read/write interrupt flag.
- 1 = A read or write operation has taken place.
- 0 = No read or write has occurred.

**FIGURE 6: INTCON REGISTER**

<table>
<thead>
<tr>
<th></th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIE</td>
<td>PEIE</td>
<td>RTIE</td>
<td>INTE</td>
<td>RBIE</td>
<td>RTIF</td>
<td>INTF</td>
<td>RBIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**PEIE**: Peripheral interrupt enable bit.
- 1 = Enables all un-masked peripheral interrupts
- 0 = Disables all peripheral interrupts.

**GIE**: Global Interrupt Enable.
- 1 = Enables all un-masked interrupts.
- 0 = Disables all interrupts.

**AUTHOR**: Scott Fink, Logic Products Division
Using the 8-Bit Parallel Slave Port

---

This program demonstrates the Parallel Slave Port function of the PIC16C64/74. The program is interrupt driven, when the PIC16CXX is either read from or written to, an interrupt is generated. If the interrupt was caused by a read, a register is incremented, and the new count is placed in an output queue. If the interrupt was caused by a write, the data is put on the Port B pins.

---

```assembly
;*************************************************************************
;* 16C64/74 Parallel Slave port
;*************************************************************************

list p=16c64,f=inhx8m
;include "c:\16C74.inc"

;Register definitions
FLAGREG equ 20h ;Flag bit register
OUTDATA equ 21h ;Output data
INDATA equ 22h ;Input data
COUNT equ 23h ;Count of times output register read

;Bit definitions for flag register
ERROR equ 00h ;Error flag bit
OUTRDY equ 01h ;Output data ready flag
INFULL equ 02h ;Input data received flag

org 0000h ;Reset Vector
goto Start

Start
    clrf OUTDATA ;Clear data registers
    clrf INDATA
    bsf STATUS,RP0 ;Select register page 1
    movlw b'00010111' ;Set RD, WR, and CS as inputs, Enable Parallel Slave port
    movwf TRIS_E
    movlw OFFh
    movwf TRIS_B ;Set Port_B to all outputs
    movlw b'10000000' ;
    movwf PIE1 ;Enable Parallel Slave Port interrupt
    bcf STATUS,RP0 ;Select register page 0
    movf OUTDATA,W ;Set output Data in PORTD
    movwf PORT_D
    movlw b'11000000' ;
    movwf INTCON

Loop
    btfsc FLAGREG,INFULL ;Check if input data received
    goto Checkout ;No data ready, check output
    bcf FLAGREG,INFULL ;Clear input data ready flag
    movf Indata,W ;Get Input data
    movwf PORT_B ;Output input data to Port_B

Checkout
    btfsc FLAGREG,OUTRDY ;Check if data output already
    goto Loop ;Not output yet, loop
    incf COUNT ;Increment output data
    movf COUNT,W ;Get output data
    movwf OUTDATA ;Put data in output queue
    bsf FLAGREG,OUTRDY ;Set flag for interrupt routine
    goto Loop
```

---

© 1994 Microchip Technology Inc.
Using the 8-Bit Parallel Slave Port

;************************************************************************************
;* Interrupt Service Routine 
;* Inputs:  FLAGREG - Flag register to/from the main routine: 
;*          Bit 0: ERROR - From Service_Int, indicates input 
;*          Bit 2: INFULL- From Service_Int, indicates data 
;*              received and in INDATA 
;*          OUTDATA - Output data queue 
;*          PIR1 - Interrupt flag register 
;*          PORT_D - Input data from slave port 
;*          TRIS_E - Parallel slave port flag register 
;* Outputs:  PORT_D - Output data to slave port 
;*          INDATA - Input data queue 
;*          FLAGREG - Flag register to/from the main routine: 
;*              Bit 0: ERROR - From Service_Int, indicates input 
;*              Bit 2: INFULL- From Service_Int, indicates data 
;************************************************************************************

Service_Int

btfss PIR1,PSPIF ;Test for Peripheral interrupt
bcf PIR1,PSPIF ;Clear Peripheral interrupt
bsf STATUS,RP0 ;Select Page 1
btfss TRIS_E,IBF ;Check if input data ready
bcf STATUS,RP0 ;Input ready, select Page 0
bsf FLAGREG,INFULL ;Set flag for main routine
movf PORT_D,W ;Get input data
movwf INDATA ;Put byte in input queue

Notinput

btfsc TRIS_E,OBF ;Check if output data read
goto Intout ;Not read, exit
bcf STATUS,RP0 ;Select Page 0
btfss FLAGREG,OUTRDY ;Check if data in output queue
goto Intout ;Output not read, exit
movf OUTDATA,W ;Get data from queue
movf PORT_D ;Put data in output buffer
bcf FLAGREG,OUTRDY ;Clear flag for main routine

Intout

bsf STATUS,RP0 ;Select Page 1
btfsc TRIS_E,IBOV ;Check input buffer overflow flag
bcf STATUS,RP0 ;Select Page 0
retfie ;Re-enable GIE and return

Error

bcf STATUS,RP0 ;Select Page 0
bsf FLAGREG,ERROR ;Set error flag for main routine
retfie ;Re-enable GIE and return

end