18F452 I/O Ports

55:036
Embedded Systems and Systems Software

I/O Ports

- Transfer data to/from pins of the microcontroller
  - Digital data (bits)
  - Analog data
  - Interrupt requests

18F452 Ports

- Five ports (total of 34 pins)
  - PORTA
    - 7 pins (RA0-RA6)
  - PORTB
    - 8 pins (RB0-RB7)
  - PORTC
    - 8 pins (RC0-RC7)
  - PORTD
    - 8 pins (RD0-RD7)
  - PORTE
    - 3 pins (RE0-RE2)

I/O Port Configuration

- Most port pins have multiple uses
  - E.g. bits 0-3 of PORTA (RA0/AN0 - RA3/AN3)
    - Digital input (RA0-RA3)
    - Digital output (RA0-RA3)
    - Analog input
      - AN0 – AN3
      - (input to on-board A/D converter)
    - AN2, AN3 can also be used for A/D reference voltage input
    - The 18F452 has a total of 8 analog inputs
  - E.g. bits 0-2 of PORTB (RB0/INT0 - RB2/INT2)
    - Digital input or output (RB0 - RB2)
    - External interrupt request (INT0 – INT2)
I/O Port Configuration--Continued

- Other special uses of I/O pins
  - External clock inputs for timers
  - Low voltage detect (analog input)
  - Capture Compare (CCP) I/O
  - SPI and I2C peripheral bus interfaces
  - USART (synchronous/asynchronous serial I/O)
  - Parallel slave port
  - etc.

Configuring the I/O Ports

- Each PORTx (x = A-E) has three associated registers:
  - PORTx (port register)
  - TRISx (data direction register)
  - LATx (used for read/modify/write operations)
    - Normally, we don’t directly access this register

PORTx Configuration

- TRISx bit settings control direction (input or output) of the corresponding PORTx bit
  - TRISx(i) = 0 ⇒ PORTx(i) configured for output
  - TRISx(i) = 1 ⇒ PORTx(i) configured for input

- Sometimes other SFRs may control aspects of Port configuration
  - E.g. for PORTA, the ADCON1 register controls analog vs. digital configuration of PORTA(3:0)
    - ADCON1(i) = 0 ⇒ PORTA(i) configured for analog input (TRISA(i) must be set for input)
    - ADCON1(i) = 1 ⇒ PORTA(i) configured for digital input/output (TRISA(i) controls the direction)

I/O Port Configuration

- Each port has some specific configuration issues—Be sure to consult the Data Sheet
  - PORTB pins have configurable weak internal pull-ups
  - Many PORTB pins can be configured to generate interrupts
    - Controlled by configuration of INTCON, INTCON2 and INTCON3 SFRs
  - Many PORTC settings are overridden by other peripheral device settings—e.g. USART, I2C
I/O Ports—Other Considerations

• RA4 is open-drain (allows “wired logic” input connections)
• RA6 is used as an oscillator input (OSC2) for the QwikFlash oscillator mode
• Most, but not all, I/O pins are Schmitt-triggered
  – This won’t be an issue for us but may be important in some applications.
• See Chapter 11 for more details and tips on external connections to I/O pins.

QwikFlash Port Usage

• The QwikFlash board has “hardwired” many I/O pins to devices on the board
  – e.g. RA4-RA1 to LEDs, RA0 to analog temperature sensor, RD7-RD4 and RE1-RE0 to LCD display, etc.
  – Some pins are unused and available on the expansion header or terminal strips

18F452 Timers

55:036
Embedded Systems and Systems Software

18Fxx2 Timer Modules

• TIMERO
  – 8 or 16 bit
  – can function as timer or counter
  – internal or external clock source
  – prescaler to adjust rate
  – timer can be read to measure elapsed time
  – can generate an interrupt on overflow (roll-over)
18Fxx2 Timer Modules--Continued

- **TIMER1**
  - 16 bits
  - Works with CCP module
  - Other features similar to TIMER0

- **TIMER2**
  - 8 bit timer
  - 8 bit period register
  - can generate interrupt when timer value matches period register value

18Fxx2 Timer Modules--Continued

- **TIMER3**
  - Similar to TIMER1

---

**Timer0 Registers**

- **TMR0L** (timer0 low byte register)
- **TMR0H** (timer0 high byte register)
- **T0CON** (Timer0 control register)

---

**T0CON Register**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>TMR0OEN: 1 = Timer0 On/0 = Timer0 Off</td>
</tr>
<tr>
<td>6</td>
<td>T0OE: 1 = Timer0 Overflow Edge Select</td>
</tr>
<tr>
<td>5</td>
<td>T0CS: 1 = T0CS Select/0 = TC0 Select</td>
</tr>
<tr>
<td>4</td>
<td>T0FME: 1 = Timer0 Overflow Mode</td>
</tr>
</tbody>
</table>
Timer0 Operation

- When Timer0 "overflows"—i.e. transitions from 11...11 to 00...00, the TMR0IF bit (bit 2) of the INTCON register is set
  - This bit must be manually reset
  - The timer continues to count

Timer0 Initialization in P1.ASM Example

Initial
MOVLF B'10001110',ADCON1 ;Enable PORTA & PORTE digital I/O pins
MOVLF B'11100001',TRISA  ;Set I/O for PORTA
MOVLF B'11101100',TRISB  ;Set I/O for PORTB
MOVLF B'10100000',TRISC  ;Set I/O for PORTC
MOVLF B'00000000',TRISD  ;Set I/O for PORTD
MOVLF B'10001000',T0CON  ;Set up Timer0
MOVLF B'00010000',PORTA  ;Turn off all four LEDs driven from PORTA

Enable
Bypass prescaler (count at instruction clock rate)
Configure as a 16 bit timer
Clock source is internal instruction cycle clock

LoopTime Subroutine from P1.ASM

Bignum equ 65536-25000+12
LoopTime
btfss INTCON,TMR0IF    ;Wait until ten milliseconds are up
bra LoopTime
movff INTCON,INTCONCOPY  ;Disable all interrupts to CPU
bcf INTCON,GIEH
movff TMR0L,TMR0LCOPY  ;Read 16-bit counter at this moment
movff TMR0H,TMR0HCOPY
movlw low Bignum
addwf TMR0LCOPY,F
movlw high Bignum
addwf TMR0HCOPY,F
movff TMR0HCOPY,TMR0H
movff TMR0LCOPY,TMR0L  ;Write 16-bit counter at this moment
movf INTCONCOPY,W      ;Restore GIEH interrupt enable bit
andlw B'10000000'
iorwf INTCON,F
bcf INTCON,TMR0IF      ;Clear Timer0 flag
return

LoopTime Subroutine from P1.ASM

Bignum equ 65536-25000+12
LoopTime
btfss INTCON,TMR0IF    ;Wait until ten milliseconds are up
bra LoopTime
movff INTCON,INTCONCOPY  ;Disable all interrupts to CPU
bcf INTCON,GIEH
movff TMR0L,TMR0LCOPY  ;Read 16-bit counter at this moment
movff TMR0H,TMR0HCOPY
movlw low Bignum
addwf TMR0LCOPY,F
movlw high Bignum
addwf TMR0HCOPY,F
movff TMR0HCOPY,TMR0H
movff TMR0LCOPY,TMR0L  ;Write 16-bit counter at this moment
movf INTCONCOPY,W      ;Restore GIEH interrupt enable bit
andlw B'10000000'
iorwf INTCON,F
bcf INTCON,TMR0IF      ;Clear Timer0 flag
return

Want to initialize the counter so that it overflows after 25,000 "ticks" = 10 msec.
LoopTime Subroutine from P1.ASM

Bignum equ 65536-25000+12+2

LoopTime
befs INTCON,TMR0IF   ;Wait until ten milliseconds are up
bra LoopTime
movff INTCON,INTCONCOPY   ;Disable all interrupts to CPU
bcf INTCON,GIEH
movff TMROL,TMROLCOPY   ;Read 16-bit counter at this moment
movlw low Bignum
addwf TMROLCOPY,F
movlw high Bignum
addwfc TMROLHCOPY,F
movff TMROLHCOPY,TMROL
movf INTCONCOPY,W   ;Restore GIEH interrupt enable bit
andlw B'10000000'
iorwf INTCON,F
bcf INTCON,TMR0IF   ;Clear Timer0 flag
return

Add Bignum to timer0 value.
Note the correction is needed because the timer is continuing to count during this period

Reinitialize timer to time remaining portion of 10 msec. interval
LoopTime Subroutine from P1.ASM

Bignum equ 65536-25000+12+2

LoopTime
  brfs INTCON,TMR0IF ; Wait until ten milliseconds are up
  bcs LoopTime
  movff INTCON,INTCONCOPY ; Disable all interrupts to CPU
  bcf INTCON,GIEH
  movff TMR0L,TMR0LCOPY ; Read 16-bit counter at this moment
  movff TMR0H,TMR0HCOPY
  movlw low Bignum
  addwf TMR0LCOPY,F
  movlw high Bignum
  addwf TMR0HCOPY,F
  movff TMR0HCOPY,TMR0H ; Write 16-bit counter at this moment
  movff TMR0LCOPY,TMR0L ; Restore GIEH interrupt enable bit
  movf INTCONCOPY,W
  andlw B'10000000'
  iorwf INTCON,F
  bcf INTCON,TMR0IF ; Clear Timer0 flag
  return

Don’t forget to do this

LoopTime Subroutine from P1.ASM

Bignum equ 65536-25000+12+2

LoopTime
  brfs INTCON,TMR0IF ; Wait until ten milliseconds are up
  bcs LoopTime
  movff INTCON,INTCONCOPY ; Disable all interrupts to CPU
  bcf INTCON,GIEH
  movff TMR0L,TMR0LCOPY ; Read 16-bit counter at this moment
  movff TMR0H,TMR0HCOPY
  movlw low Bignum
  addwf TMR0LCOPY,F
  movlw high Bignum
  addwf TMR0HCOPY,F
  movff TMR0HCOPY,TMR0H ; Write 16-bit counter at this moment
  movff TMR0LCOPY,TMR0L ; Restore GIEH interrupt enable bit
  movf INTCONCOPY,W
  andlw B'10000000'
  iorwf INTCON,F
  bcf INTCON,TMR0IF ; Clear Timer0 flag
  return

Don’t worry about this stuff for now.

TIMER1

• TIMER1 is a 16-bit timer:
  TMR1H - MSB of current time
  TMR1L - LSB of current time

• TIMER1 prescaler allows:
  1 (no prescaling)
  2 (2 clock cycles per timer increment)
  4
  8

TIMER1 Configuration (T1CON)

M.1 T1CON: Timer1 Control Register

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M.1</td>
<td>T1CON: Timer1 Control Register</td>
</tr>
</tbody>
</table>
### Resolution vs. Max. Time

- **TIMER1** (internal) clock (unscaled) is instruction clock—i.e. \( \frac{F_{osc}}{4} \)
- Assuming a 10 MHz oscillator \( (F_{OSC} = 10 \text{ MHz}) \) and not using the PLL (internal clock = 2.5 MHz \( \rightarrow \) period = \( \frac{1}{2.5M} = 0.4 \mu s \)):

<table>
<thead>
<tr>
<th>Prescale</th>
<th>Resolution</th>
<th>Max. Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \times 1 )</td>
<td>0.4 ( \mu s )</td>
<td>26 ms</td>
</tr>
<tr>
<td>( \times 2 )</td>
<td>0.8 ( \mu s )</td>
<td>52 ms</td>
</tr>
<tr>
<td>( \times 4 )</td>
<td>1.6 ( \mu s )</td>
<td>105 ms</td>
</tr>
<tr>
<td>( \times 8 )</td>
<td>3.2 ( \mu s )</td>
<td>210 ms</td>
</tr>
</tbody>
</table>

### Timer Example

- To time a bit period (9600 baud) \( \approx 104.167 \text{ microseconds} \)
- Assuming a 10 MHZ oscillator
- Requires: \( 104.167/0.4 = 260.4175 \) (instruction) cycles
- So count value to be loaded into the timer (to cause overflow after 260 cycles) is:
  \[ 2^{16} - 260 = 65,276 \] (not including any additional correction factors)

### TMR1H Buffer

- Potential problem caused by 16-bit timers, but only 8 bit register transfer capability:

  ```
  movf TMR1L, W ; TMR1 = 0x03FF
  movwf LT
  movf TMR1H, W ; TMR1 = 0x0401
  movwf HT
  ```

  Result is HT = 0x40 and LT = 0xFF, so the observed time is 0x04FF, which is wrong.

- To fix problem, TMR1H is not the actual MSB of the timer, but a buffer which gets a new value whenever TMR1L is read:

  ```
  movff TMR1L, LT ; TMR1 = 0x03FF
  movff TMR1H, HT ; TMR1 = 0x0400
  ```

  \( LT \leftarrow 0xFF \) and \( HT \leftarrow TMR1H \) (buffer) \( \leftarrow 0x03 \)
**TMR1H Buffer**

- TMR1H buffer is also written to the actual timer when TMR1L is written.
- To correctly read Timer 1: Read TMR1L first, then read TMR1H
- To correctly write Timer 1: Write TMR1H first, then write TMR1L
- Note: To get this buffered behavior for correct reading and writing of the timer, the RD16 bit (bit 7) of the TICON (Timer 1 Control) register must be set.

---

**Timer 1 Basic Operation**

- RD16 = 1 to enable the TMR1H buffer.
- TMR1ON = 1 to make timer increment when input clock ticks.
- T1CKPS1 and T1CLPS0 = 00 (÷1), 01 (÷2), 10 (÷4), 11 (÷8).

---

**T1CON Register**

- T1CON = RD16, X, T1CKPS1, T1CKPS0, T1OSCEN, T1SYNC, TMR1CS, TMR1ON
- RD16 = 1 to enable the TMR1H buffer.
- TMR1ON = 1 to make timer increment when input clock ticks.
- T1CKPS1 and T1CLPS0 = 00 (÷1), 01 (÷2), 10 (÷4), 11 (÷8).
Timer 1 Interrupt on Overflow

• When Timer 1 changes from 0xFFFF to 0x0000 (an overflow), the timer sets the TMR1IF bit of the PIR1 register (always).
• If the TMR1IE bit of the PIE1 register is set, this interrupt source is enabled and can potentially cause an interrupt when Timer 1 overflows.
• If the IPEN bit of the RCON register is set, then the PIC is using 2-level interrupts (high and low) and we can choose whether Timer 1 generates high priority or low priority interrupts.

Timer 1 Interrupt on Overflow

• If the TMR1IP bit of the IPR1 register is set, then we have chosen high priority for Timer 1 overflow interrupts (TMR1IP = 0 chooses low priority).
• If the GIEH bit of the INTCON register is reset, then all interrupts are turned off. IF GIEH is set, then high-priority interrupts are on and low-priority interrupts depend on GIEL.
• If the GIEL bit of the INCON register is reset, then all low-priority interrupts are off.

Figure 13.7 Timer 1 for encoded time-interval measurements.